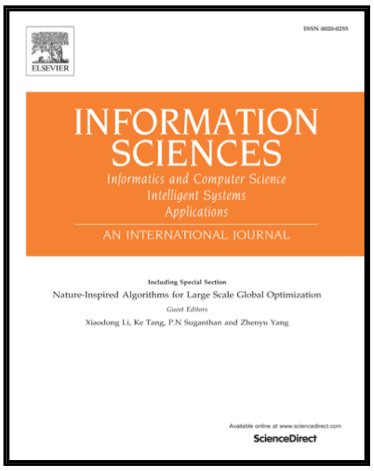


An Improved Differential Evolution Algorithm using Efficient Adapted Surrogate Model for Numerical Optimization

Noor H. Awad , Mostafa Z. Ali , Rammohan Mallipeddi , Ponnuthurai N. Suganthan

PII:	S0020-0255(18)30280-9
DOI:	10.1016/j.ins.2018.04.024
Reference:	INS 13567
To appear in:	*Information Sciences*
Received date:	26 November 2017
Revised date:	19 February 2018
Accepted date:	3 April 2018

Please cite this article as: Noor H. Awad , Mostafa Z. Ali , Rammohan Mallipeddi , Ponnuthurai N. Suganthan , An Improved Differential Evolution Algorithm using Efficient Adapted Surrogate Model for Numerical Optimization, *Information Sciences* (2018), doi: 10.1016/j.ins.2018.04.024

An Improved Differential Evolution Algorithm using Efficient Adapted Surrogate Model for Numerical Optimization

Names and affiliations of all authors:

(1) Noor H. Awad, School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore 639798, noor0029@e.ntu.edu.sg
(2) Mostafa Z. Ali, Computer Information Systems department, Jordan University of Science & Technology, Jordan, mzali.pn@gmail.com
(3) Rammohan Mallipeddi, School of Electronics Engineering, Kyungpook National University, South Korea, mallipeddi.ram@gmail.com
(4) Ponnuthurai N. Suganthan, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, epnsugan@ntu.edu.sg

Contact information of corresponding authors:

1. Post: Computer Information Systems, Jordan University of Science & Technology, Irbid, Jordan, Email: mzali.pn@gmail.com, Phone: +(962) 003-6901
2. Post: School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, Email: epnsugan@ntu.edu.sg, Phone: +(65) 67905404
3. Post: Rammohan Mallipeddi, School of Electronics Engineering, Kyungpook National University, South Korea, mallipeddi.ram@gmail.com, Phone: +(82) 1035751276

Abstract

Contemporary real-world optimization benchmarks are subject to many constraints and are often high-dimensional problems. Typically, such problems are expensive in terms of computational time and cost. Conventional constraint-based solvers that are used to tackle such problems require a considerable high budget of function evaluations. Such budget is not affordable in practice. In most cases, this number is considered the termination criterion in which the optimization process is stopped and then the best solution is marked. The algorithm might not converge even after consuming the pre-defined number of function evaluations, and hence it does not guarantee an optimal solution is found. Motivated by this consideration, this paper introduces an effective surrogate model to assist the differential evolution algorithm to generate competitive solutions during the search process. The proposed surrogate model uses an adaptation scheme to adapt the *theta* parameter in the Kriging model. This variable determines the correlation between the parameters of the problem. For that reason, an accurate surrogate model is crucial to have a noticeable enhancement during the search. The statistical information exploited from a covariance matrix is used to build the correlation matrix to adapt the *theta* variable instead of using a fixed value during the search. Hence, the surrogate model evolves over the generations to better model the basin of the search, as the population evolves. The model is implemented in the popular L-SHADE algorithm. Two benchmark sets: bound-constrained problems and real-world optimization problems are used to validate the performance of the proposed algorithm, namely iDE*a*Sm. Also, two engineering design problems are solved: welded beam and pressure vessel. The performance of the proposed work is compared with other state-of-the-art algorithms and the simulation results indicate that the new technique can improve the performance to generate better statistical significance solutions.

Keywords: *evolutionary algorithm, differential evolution, surrogate model, kriging model*

1. Introduction

Many evolutionary algorithms (EAs) have been developed as population-based algorithms which use operations that are inspired from the natural evolution process to solve a variety of optimization problems [2, 44, 45]. Without loss of generality, an optimization problem can be mathematically expressed as follows:

$$\text{minimize} \quad f(\vec{X}),\ (f:\Omega \subseteq \Re^{D} \rightarrow \Re) \tag{1}$$

where $f(\vec{X})$ which involves a vector $\vec{X} = [x_1, x_2, ..., x_D]^T$ of D decision variables is an objective function. The task of an optimization algorithm is to find the optimal solution, $\vec{X}^*$ that minimizes $f(\vec{X})$ such that $f(\vec{X}^*) < f(\vec{X}), \forall \vec{X} \in \Omega$ where Ω denotes the search domain.

One of the major differences that is noticed between the developed evolutionary algorithms is in the reproduction operator [2]. This operator defines how the new trial solutions are generated and evolved during the optimization process. Among these algorithms, the differential evolution (DE) is a stochastic metaheuristic algorithm which has shown noticeably good performance in solving a variety of optimization problems [39, 7]. This algorithm constitutes three basic steps: mutation, crossover and selection that are used iteratively with a given budget of function evaluations (FEs), trying to improve the found solution value during the search. It is known that DE is dependent on the settings of its control parameters which reflects its performance [26]. Those control parameters are: the population size (*NP*), the scaling factor (*F*), and the crossover rate (*CR*). In the early studies, the trial and error method was suggested to control their settings which is a time consuming process and hence is not efficient, and may not lead to better solutions [7]. In the last few years, many studies developed different ways to control the manner in which the values of *F* and *CR* are set using adaptive and self-adaptive methods [7]. The key idea is to encode the parameter values within each individual and then evolve them during the search. During this evolution, the values which are capable of generating better offspring solutions are likely to survive based on the performance feedback on used control parameters values during the search. On the other hand, few studies suggested to dynamically modify the population size to reflect good performance during the evolution process [7].

In the DE algorithm, like any other evolutionary algorithm, the optimization process continues until the termination criterion is met which is defined after consuming a pre-defined number of FEs. Increasing this number ultimately helps the algorithm to generate better solutions if the algorithm has not converged yet. To achieve this, approximations can be used in the optimization algorithm to reduce the number of FEs [34]. The goal is to approximate the objective function using certain models. Those approximations are referred to as surrogate models which can be used to evaluate the new trial solutions and hence they are computationally inexpensive when compared to the original objective function [3]. The history of population individuals in an evolutionary algorithm can be used to build a surrogate model which aims to reduce the FEs and hence assist its evolution to generate better solutions [18, 11]. Building a surrogate model involves the use of a function approximation to estimate the response of the original function through training using two sets, (X, y) where $X = \{X_1, X_2, ..., X_N\}$ is a set of vectors (solutions) of D decision variables and y is the responses (functions values). Hence, the surrogate model is expressed as $y = f(X)$ which estimates the original objective function. Different surrogate models have been proposed in the literature such as: radial basis function networks, polynomials, artificial neural networks and Kriging [18].

In this paper, the main target is not to propose a new differential evolution algorithm, but to propose a new approach to improve its performance and generate new and better solutions using an effective surrogate model. The key advantage of the proposed surrogate model in this work is in the use of a novel scheme to adapt the *theta* parameter to build an accurate model to approximate the response of an objective function in an effective manner. This adaptation scheme uses the information exploited from the computed covariance matrix of the current population which is then converted to a correlation matrix. This correlation matrix can be used to adapt the *theta* parameter during the search instead of using one fixed value. Using this way, the accuracy of the used surrogate model is enhanced to mimic the response of the original objective function. The proposed surrogate model is used to assist the well-known L-SHADE algorithm [42] to produce better solutions by using more than one chance to generate better offspring during the search process. Therefore, the probability to generate competitive offspring solutions is increased. Moreover, the proposed surrogate model is used to enhance the best solution if a stagnation case is caught using an auxiliary local search procedure. On the other hand, the time to build the surrogate model can be considered an issue which adds extra computational cost. The new algorithm, an improved differential evolution with an adapted surrogate model, namely iDE*a*Sm, is validated using two different benchmarks suites. The first one is a set of bound-constrained optimization problems and the second benchmark suite constitutes a set of real-world optimization problems, which is also used to test the validity of the introduced surrogate model to estimate the response of challenging real-life applications. The simulation results show that the proposed algorithm is able to generate good solutions as compared to other algorithms. Also, two well-known engineering design problems are used to test the proposed algorithm which are welded beam and pressure vessel. The obtained results prove the effectiveness and the capability of the proposed algorithm in solving different types of problems with variant characteristics and complexity.

The rest of the paper is organized as follows. Section 2 introduces the differential evolution algorithm and surrogate model concept. Section 3 reviews the literature review and summarizes the use of related surrogate models to assist the performance of DE. The proposed algorithm is introduced in Section 4. The experimental results and analysis for three experiments are presented in Section 5. Finally, the conclusion is summarized in Section 6.

2. Preliminaries

2.1. Differential Evolution

DE algorithm is one of the simplest evolutionary algorithms, yet effective, which has proven the quality of its performance to solve a variety of optimization problems [39]. In this algorithm, four basic steps are needed to evolve the individuals over the generations: initialization, mutation, crossover and selection. A population of *NP* random individuals is initialized within the lower and upper bounds of the problem being solved as shown below:

$$x^j_{i,g_0} = x^j_{\min} + rand(0,1).(x^j_{\max} - x^j_{\min}) \quad j = 1, 2, ..., D \tag{2}$$

where *i* is the index of an individual X_i which is composed of *D* parameters that are uniformly created within the lower and upper bounds of the search range which are represented as: $X_{\min} = (x^1_{\min}, ..., x^D_{\min})$ and $X_{\max} = (x^1_{\max}, ..., x^D_{\max})$. $rand(0,1)$ is a random generator between 0 and 1.

Next, the initial population is evaluated by computing the function value for each individual. The optimization loop is then started to evolve the individuals and generate new offspring values using: mutation, crossover and selection. The mutation step generates new mutant vectors $V_{i,g} = (v^1_{i,g}, v^2_{i,g}, ..., v^D_{i,g})$ using one of the mutation strategies as listed in Eqs. (3)-(7), where *F* is the scaling factor and r_1, r_2, r_3, r_4, r_5 are the indices of mutual random individuals which are chosen from the current population:

DE/rand/1: $V_{i,g} = X_{r_1,g} + F \cdot (X_{r_2,g} - X_{r_3,g})$ (3)

DE/best/1: $V_{i,g} = X_{best,g} + F \cdot (X_{r_1,g} - X_{r_2,g})$ (4)

DE/current-best/1: $V_{i,g} = X_{i,g} + F \cdot (X_{best,g} - X_{i,g} + X_{r_1,g} - X_{r_2,g})$ (5)

DE/best/2: $V_{i,g} = X_{best,g} + F \cdot (X_{r_1,g} - X_{r_2,g} + X_{r_3,g} - X_{r_4,g})$ (6)

DE/rand/2: $V_{i,g} = X_{r_1,g} + F \cdot (X_{r_2,g} - X_{r_3,g} + X_{r_4,g} - X_{r_5,g})$ (7)

The crossover operation is applied after the mutation step to combine the mutant vector, $V_{i,g}$, and the target vector, $X_{i,g}$, in a final offspring $U_{i,g}$ using the binomial crossover as shown in Eq. 8, where *CR* is the crossover rate and j_{rand} is a random integer index in the range [1, *D*]. The parameter values are copied from $V_{i,g}$ with a probability of *CR*, $(rand \le CR)$, or if the j^{th} index is the same as j_{rand} to guarantee that at least one of the parameters is copied from the mutant vector. Otherwise, the parameter values of $U_{i,g}$ are copied from $X_{i,g}$.

$$u^j_{i,g} = \left\{ \begin{array}{l} v^j_{i,g}, if\ (with\ probability\ of\ CR)\ or\ (j = j_{rand}) \\ x^j_{i,g}, otherwise \end{array} \right\} j = 1, 2, ..., D \tag{8}$$

After that, the selection operation takes place to determine whether the target vector $X_{i,g}$ is replaced by the new offspring $U_{i,g}$ based on the function values as shown in Eq. 9. The better individuals will survive to the next generation to evolve again using the same operations until the termination criterion is met.

$$X_{i,g+1} = \begin{cases} U_{i,g}, & \text{if } f(U_{i,g}) \le f(X_{i,g}) \\ X_{i,g}, & \text{otherwise} \end{cases} \tag{9}$$

2.2. Surrogate Model

A surrogate model is basically a mathmetical model. It is considered as a statistical model of the response surface of a simulation model. To estimate the behavior of an objective function, the surrogate model is used which can be defined as a meta-model using the mathematical functions by studying the relationship between (x, y) where x is the input samples and y is the output response to be presented as:

$$y = f(\mathrm{X}) \tag{10}$$

The approximation form of the above equation is the surrogate model as shown below:

$$\hat{y} = \hat{f}(\mathrm{X}) \tag{11}$$

where the output of the surrogate model mimics the original objective function in such a manner as $\hat{y} = y + \varepsilon$.

Different procedures have been introduced to build the surrogate models such as: support vector machines (SVM), response surface methods (RSM), Kriging, artificial neural networks (ANN). In [4, 37], a comparison between the different methods to build a surrogate model is introduced. Those studies used small set of test problems and considered the use of only one criterion to select the best model. On the other hand, the authors in [11] demonstrated a study on the usage of surrogate models and how to select an appropriate model for the evolutionary algorithms based on two factors: suitability of combining the surrogate model in an evolutionary algorithm and dimensionality. The authors of the work in [11] recommend the use of support vector regression method or Kriging model as the best options to estimate the optimization problem with low to moderate dimensions. Moreover, they concluded that the radial basis function network is the best option to model high dimensionality problems.

The Kriging models have been used in different applications for numerical and engineering problems [33]. It also became popular in the last decade to be used to improve the performance of an evolutionary algorithm to solve many benchmark sets including expensive optimization problems [11].

The function of interest, $y(\mathrm{x})$ in Kriging model can be expressed as shown in Eq. 12 using two models: global and localized to better interpolate the input data:

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

The polynomial function, $f(\mathrm{x})$ defines the global definitions of actual function, and $Z(\mathrm{x})$ is a representation of the localized deviations that represents the stochastic process using a Gaussian model with a mean equals to 0 and variance of σ^2. The covariance of $Z(\mathrm{x})$ is presented in the following equation:

$$\mathrm{Cov}\,[\mathrm{Z}(\mathrm{x}^i),\ \mathrm{Z}(\mathrm{x}^j)] = \sigma^2 \Re(\theta, \mathrm{x}^i, \mathrm{x}^j) \tag{13}$$

where $\Re$ is the correlation matrix for a data set of samples. Given two samples, *i* and *j*, the correlation between them is represented as $\Re(\theta, \mathrm{x}^i, \mathrm{x}^j)$.

3. Literature review

Many differential evolution algorithms have been proposed in the literature [7]. Most of these algorithms have used adaptive or self-adaptive methods to control the parameter settings over the generations to guide the evolutionary search to the promising regions. The adaptive methods rely on the information feedback during the search to select suitable settings based on the successful generation of new offspring values. JADE is one of the well-known DE adaptive methods [48]. This algorithm became later a base for many other researchers to build their improved algorithms such as: SHADE [41] and L-SHADE [42]. The "current-to-*p*best" was introduced as a new mutation strategy in JADE where it uses the top *p* best individuals to select the best individual, X_{pbest}, to guide the evolution. The reason behind using this strategy is to alleviate the stagnation and premature convergence that greedy strategies such as "current-to-best" causes over the generations. The Cauchy and normal distributions are used to adapt *F* and *CR* for each individual with means that are equal to μ_F and μ_{CR}, respectively, and a value of 0.1 is used for the standard deviation. The two mean values are updated using the stored successful values of *F* and *CR* which are responsible for generating a new offspring that is better than its parent at the end of each generation. SHADE algorithm [49] was proposed after that. SHADE adds a history-based scheme that consists of a two dimensional memory to randomly select one of the mean values. The used memory is initialized with 0.5 and then the k^{th} position is updated using the weighted Lehmer mean for the successful *F* and *CR* value during the generation. The same authors of SHADE proposed a new improved version that is called L-SHADE [42]. This algorithm showed a superior performance to solve the IEEE CEC'14 single objective benchmark problems. In L-SHADE, the same adaptation scheme for *F* and *CR* was used, in addition to a linear dynamic reduction to change the size of the population space at every generation. The algorithm starts with a large population size, then a reduction scheme is used to reduce the size over the generations to reach a pre-defined minimum size.

Other concepts are also introduced to improve the performance of DE. In [47], the authors aim to enhance the diversity of the population using an auto-enhanced mechanism which is activated when a stagnation case is caught. The algorithm aims to provide better solutions through diversifying the population for each dimension level to avoid reaching a stagnation scenario. A new crossover strategy which is based on multiple recombination was introduced recently in [32]. The authors aimed to exploit the key advantages of the existing crossover types to propose a new scheme to handle the limitations of having dependent variables. A multiple variant coordination framework was proposed in [49]. In their introduced technique, the evolutionary search was divided into two non-overlapping segments using two mechanisms: preserving mechanism and selecting mechanism. The goal is to use different search strategies and then select the best optimizer to be utilized independently for each segment. An improved orthogonal learning strategy is used for crossover and selection operations [24]. The algorithm utilizes the orthogonal experimental design to improve the discovery of information at each dimensional level during the evolutionary search. A new extension of JADE algorithm was recently proposed in [50], where the authors, as opposed to JADE algorithm, sort the generated crossover values, and small values are assigned to the individuals with better function values. Such mechanism aimed at increasing the possibility of copying the parameter values which have better fitness values so they are propagated to the following generations.

Many surrogate-assisted differential evolution algorithms have been proposed in the literature [18]. The main goal was to use a well-designed model to mimic the response of an objective function, for the optimization problem being solved, to be used to evaluate the trial vectors. Based on that, the function evaluations using the original objective function is reduced. Most of these algorithms deal with expensive optimization problems from the point of view that a limited number of function evaluations is allowed for an expensive optimization problem. On the other hand, the surrogate-assisted algorithms have been used to solve practical problems as in [25] and also for multi-objective optimization problems [20]. Sa-DE-DPS algorithm was proposed in [20], where the authors used a simple Kriging model which was built using the initial population samples. Then, new generation of trial vectors were generated using different mutation strategy which was selected in a random fashion. Those new trial vectors were then evaluated using the surrogate model. The best individual was selected to be evaluated using the original objective function and then it was added to the samples to rebuild the surrogate model for the next generation. HSBA algorithm was proposed in [38], in which the authors used three stages to solve a benchmark set of expensive optimization problems. The first stage was the global search in which the DE operations were used for 10 generations using the actual objective function. The stored samples in those 10 generations were then used to build a Kriging model to be utilized later to evaluate the trial vectors. In the third stage, the quality of the best solution was enhanced using local search. The authors in [14], used a cheap surrogate model based on density estimation PSO and DE algorithm. The surrogate model was built using Parzen window method which used the kernel density estimation, and hence the density for each sample point was approximated. ESMDE algorithm was proposed in [33] which was a new version of EPSDE algorithm using an evolving surrogate model. Original EPSDE used an ensemble of different mutation strategies to guide the evolutionary search using a randomly generated number for *F* and *CR* [29]. The authors in [29], extended this algorithm by adding a surrogate model to help evaluating the different offspring without using the original function.

4. Proposed Algorithm

The proposed algorithm, namely iDE*a*Sm, is introduced and explained in this section. A well-designed surrogate model is incorporated with L-SHADE to improve its performance. In the following sub-sections, a review of L-SHADE algorithm is presented and then iDE*a*Sm is explained in details with the use of the surrogate model.

4.1. A Review of L-SHADE Algorithm

L-SHADE algorithm uses the same concepts which have been introduced in JADE and SHADE algorithms, and adds a linear population size reduction to dynamically change the population size at each generation [42]. In this algorithm, a large size is used at the beginning of the search, and initializes a set of random individuals using Eq. 2 within the search range of the problem being solved. Then, "current-to-*p*best" is performed to generate new mutant vectors which uses one of the top *p* best individuals to guide the evolutionary search. In this mutation strategy, as shown in Eq. 14, $X_{r_1,g}$ is chosen randomly from the population space with an index $r_1, r_1 \in [1, NP]$, while an external archive, A, is used as proposed originally by JADE to select $X_{r_2,g}$, where the index r_2 is chosen from the union of the current population and the archived one at generation *g*. At the first generation, the used archive is filled with the initial population, and then it is filled with the inferior individuals at every other generation. In case the archive is full and to make space for the newly added individuals, randomly selected individuals are deleted from the archive.

$$V_{i,g} = X_{i,g} + F_{i,g}.(X_{pbest,g} - X_{i,g}) + F_{i,g}.(X_{r_1,g} - X_{r_2,g}) \tag{14}$$

A history-based scheme as proposed by SHADE is used again in L-SHADE algorithm to adapt the scaling factor and crossover rates, $F_{i,g}$ and $CR_{i,g}$, respectively. The Cauchy and normal distributions are used with mean values, μF_{r_i} and μCR_{r_i}, respectively, and a standard deviation that equals to 0.1 as shown in Eq. 15 and 16 to generate $F_{i,g}$ and $CR_{i,g}$:

$$F_{i,g} = randc(\mu F_{r_i}, 0.1) \tag{15}$$

$$CR_{i,g} = randn(\mu CR_{r_i}, 0.1) \tag{16}$$

A two dimensional memory M is used to store the mean values μF and μCR. The memory is initialized with 0.5 at the first generation. Then the successful values for $F_{i,g}$ and $CR_{i,g}$ which are able to generate an offspring value that is better than its parent are stored in S_F and S_{CR}, to be used to update the memory with an index k. This index is set to 1 at the beginning of the search, and is then incremented at each generation to update the mean values using the weighted Lehmer mean, $mean_{WL}$ as shown in the following equations.

$$\mu F_{k,g+1} = mean_{WL}(S_F) \tag{17}$$

$$\mu CR_{k,g+1} = mean_{WL}(S_{CR}) \tag{18}$$

$$mean_{WL}(S) = \frac{\sum_{k=1}^{|S|} w_k . S_k^2}{\sum_{k=1}^{|S|} w_k . S_k} \tag{19}$$

$$w_k = \frac{\Delta f_k}{\sum_{j=1}^{|S|} \Delta f_j} \tag{20}$$

$$\Delta f_k = | f(U_{k,G}) - f(X_{k,G}) | \tag{21}$$

At the end of each generation, the population size NP is linearly decreased, where the worst-ranked individuals are eliminated to set the new computed population size as shown below. $NP_{\min}$ and $NP_{\max}$ are set to 4 and $18 \times D$, respectively, FEs and $FEs_{\max}$ are the current and maximum number of FEs.

$$NP(g+1) = Round[(\frac{NP_{\min} - NP_{\max}}{FEs_{\max}}).FEs + NP_{\max}] \tag{22}$$

4.2. The Proposed Adapted Surrogate Model

In the proposed algorithm, iDE*a*S, an adapted surrogate model is used to help L-SHADE to generate better solution during the evolutionary search. In this work, the Kriging model is used due to the following reasons [3, 47]:

- Its simplicity and capability to estimate the objective function.
- The spatial and temporal correlation of the input data used to estimate the objective function and its shape.
- A good estimation with a satisfactory confidence interval can be obtained with a reasonable computational cost compared to other models.

To build the Kriging model, input samples of $\{X, f(X)\}$ are needed. Those samples are required to use the original function $f(X)$ to evaluate an input X. Then, a mathematical model, as explained in section 2.2, is used to interpolate the samples. As shown in Eq. 13, the Kriging model depends mainly on the correlation between the samples, denoted as theta parameter θ, to estimate the response. In the literature, the algorithms which used the Kriging model have used a fixed value for θ parameter, and this value is set in an arbitrary way using one value for all the generations. In this proposed algorithm, an adaptation scheme is used to adapt the correlation over the generations to build a well-designed Kriging model. This scheme falls in the first class which uses the information feedback from the input samples to set the θ parameter, which reflects the actual correlation between the current samples.

The current population at generation *g* with the function values for its individuals represent the input samples for our Kriging model as shown below:

$$\begin{aligned} &\{X_1, X_2, \dots, X_{NP}\} \\ &\{f(X_1), f(X_2), \dots, f(X_{NP})\} \end{aligned} \tag{23}$$

In particular, the extracted information from the population such as covariance, variance and mean values can be utilized to represent its statistical properties. The information can be extracted by computing the covariance matrix. The key feature that the covariance matrix can provide is the interaction between the variables. Hence, it should be systematically very useful to utilize this matrix to compute the correlation between the variables. Motivated by this consideration, the covariance matrix learning is used in this paper to adapt the θ parameter in the proposed Kriging model. The procedure of θ adaptation is introduced as follows:

Step 1. Use the current population to compute the covariance matrix, and then apply the Eigen decomposition as shown in the following equation using the orthogonal matrices: B^T, B and diagonal matrix D_m.

$$C = BD_m B^T \tag{24}$$

The Eigen vector of the covariance matrix *C* represents each column in the matrix B.

Step 2. Compute the square roots of the diagonal variances of *C*, denoted as the matrix D_m, as shown below:

$$\delta = \sqrt{D_m} \tag{25}$$

Step 3. Use the covariance matrix *C* and δ to compute the correlation matrix *E* by standardizing each row and column of *C* as follows:

$$E = \frac{C}{\delta \times \delta'} \tag{26}$$

The correlation matrix *E* contains the correlation coefficients between each variable and the others.

Step 4. Set the θ parameter to the mean value of the upper triangular coefficients of the correlation matrix *E* as follows:

$$\theta = mean(\widehat{E}) \tag{27}$$

Note that, the correlation matrix *E* is a symmetric matrix and hence the upper triangular coefficients denoted as $\widehat{E}$ are taken to compute the mean.

4.3. Putting it all together: iDE*a*Sm algorithm

The proposed adapted surrogate model, which is explained in Section 4.2, is incorporated in L-SHADE to help assist the DE evolution to generate better offspring in each generation. A flowchart of the proposed algorithm is presented in Fig. 1.

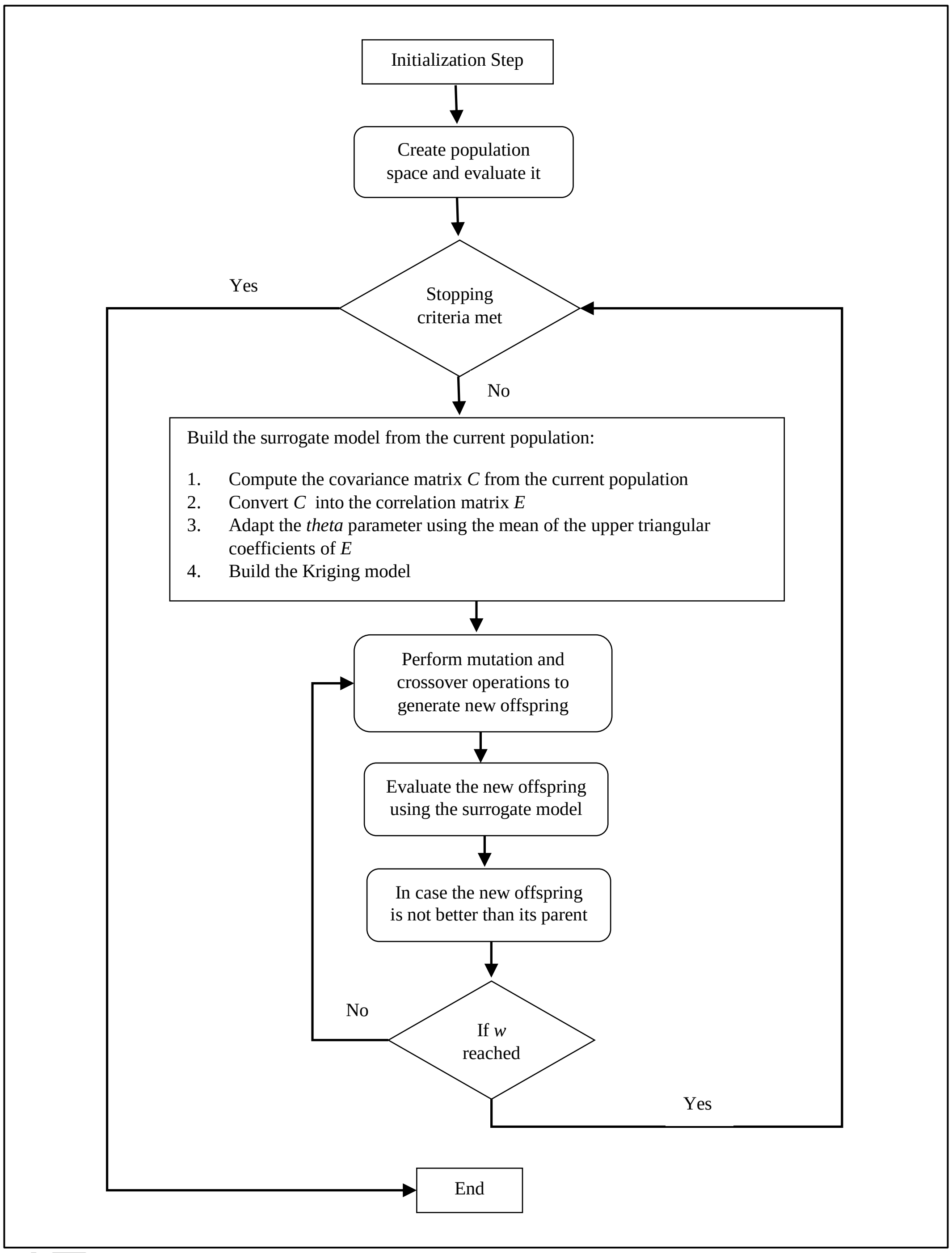

Initialization Step
Create population space and evaluate it
Stopping criteria met
Yes
No
Build the surrogate model from the current population:
1. Compute the covariance matrix C from the current population
2. Convert C into the correlation matrix E
3. Adapt the theta parameter using the mean of the upper triangular coefficients of E
4. Build the Kriging model
Perform mutation and crossover operations to generate new offspring
Evaluate the new offspring using the surrogate model
In case the new offspring is not better than its parent
If w reached
No
Yes
End

Fig. 1. A flowchart of iDE*a*Sm algorithm

The algorithm starts with the initialization step for the used parameter, next, a population space is created and evaluated. After that, the surrogate model is built after the θ parameter is adapted using the computed coefficient matrix of the current population. After that, the mutation and crossover operations are performed, and the new generated offspring is evaluated using the surrogate model. If the new offspring is not better than its parent, then another chance is given to generate new offspring which is evaluated using the surrogate model until the maximum chances are reached denoted as *w*.

A full pseudo-code of iDE*a*Sm is presented in Fig. 2. After the initial population is generated, each individual is evaluated by computing its function value as shown in Lines 1-2 by calling the original function for the problem being solved. Then, at the beginning of each optimization loop, the proposed surrogate model is constructed using the current population as an input samples denoted as $\{X_1, X_2, ..., X_{NP}\}, \{f(X_1), f(X_2), ..., f(X_{NP})\}$. The θ parameter is adapted to build Kriging model, as explained in section 4.2, in which the covariance matrix *C* is computed from the current population, and converted into the correlation matrix *E*. Next, the mean value of the upper triangular coefficients of this matrix is used to adapt the θ parameter. $y(\mathrm{x})$ as the function of interest is computed as shown in Eq. 12 and 13. Using this adaptation scheme which utilizes the actual interactions between the current samples by using the covariance and correlation matrices help to build a well-designed surrogate model which reflects a good estimation to the original function.

After the surrogate model is built, a new offspring is generated using the mutation and crossover operations as introduced in section 4.1. The surrogate model is used to evaluate this new offspring utilizing the function of interest, $y(\mathrm{x})$ which mimics the original function being solved. In case that the new offspring is not better than its parents, different chances are given to generate other offspring using different values for *F* and *CR* until a competitive offspring is generated or a maximum number of chances, *w*, is reached, as shown in Lines 9-21. This way, the proposed surrogate model is utilized to generate good offspring values at every generation, and hence assist the evolutionary search to generate competitive solutions. The new competitive offspring values are then evaluated using the original objective function as shown in Lines 22-28. The purpose is to calculate the actual function values in which they will be used to re-build the surrogate model for the next generation. Replacement occurs in case the new generated offspring has a better function value, and the successful values of *F* and *CR* are stored to update the memory as shown in Line 29.

The built surrogate model is also used to enhance the quality of the best solution at each generation using a local search method, based on interior point method as shown in Lines 30-36. Firstly, the best solution is picked and is denoted as $X_{best,g}$. Then, the function of interest $y(\mathrm{x})$ is used by the interior point method [30] to generate a new solution. Each time a new solution is generated by the interior point method, the surrogate model is used to evaluate this solution. Finally, the final solution is evaluated using the original objective function to replace the best solution if it generates a better function value. In this manner, the best solution is enhanced by utilizing the surrogate model without using the original function and hence the FEs used by this local search are not calculated.

At the final stage of iDE*a*Sm, the linear population size reduction is called as shown in Lines 38-41 to dynamically reduce the population size using Eq. 22. The individuals are sorted based on their function values, and the worst ranked individuals are eliminated from the population. The surrogate model is re-built using the new population space with its new individuals. This is used to better mimic the actual objective function using the proposed adaptation for θ parameter which reflects a new correlation value for the new individuals. At this point, a new optimization loop is repeated again which utilizes the new surrogate to generate new competitive solutions until the termination criterion is met.

```
Algorithm: iDEaSm
1. Initialize population at first generation $g_0$, $P_{g_0} = \langle x_1^{g_0}, \ldots, x_{NP}^{g_0} \rangle$ as shown in Eq. 2.
2. Evaluate the initial population by computing the function values for each individual: $\{f(X_1), f(X_2), \ldots, f(X_{NP})\}$
3. Initialize memory with 0.5 for both $\mu F$ and $\mu CR$
4. While termination criterion is not met Do
5.   Build the adapted surrogate model
6.     Use $\{X_1, X_2, \ldots, X_{NP}\}, \{f(X_1), f(X_2), \ldots, f(X_{NP})\}$ as an input to Kriging model
7.     Use steps 1-4 as explained in section 4.2 to adapt the $\theta$ parameter
8.     Compute the function of interest as shown in Eq. 12 and 13
9.   For $i$=1to $NP$
10.    $q$ = 1
11.    While $q > w$ or $flag == true$
12.      Generate $F_{i,g}$ and $CR_i$ using Cauchy and normal distributions as shown in Eq. 15 and 16
13.      Generate new mutant vector $V_{i,g}$ using Eq. 14
14.      Apply the binomial crossover to generate $U_{i,g}$
15.      Evaluate the new offspring $y(U_{i,g})$ using the surrogate model
16.      If $y(U_{i,g}) < f(X_{i,g})$
17.        $flag = true$
18.      EndIf
19.      $q = q + 1$
20.    EndWhile
21.  EndFor
22.  For $i$=1to $NP$
23.    Evaluate $U_{i,g}$ using the original objective function
24.    If $f(U_{i,g}) < f(X_{i,g})$
25.      $X_{i,g} = U_{i,g}$
26.      Save the successful values for $F_i$ and $CR_i$ in $S_F$ and $S_{CR}$ respectively
27.    EndIf
28.  EndFor
29.  Use Eqs. 17-21 to update the memory
30.  Call local search
31.    Pick the best solution, $X_{best,g}$
32.    Perform the interior point method to generate new solution ($T$) and evaluate it using the surrogate model, $y(T)$
33.    Evaluate the final solution using the original function value $f(T)$
34.    If $f(T) < f(X_{best,g})$
35.      $X_{best,g} = T$
36.    EndIf
37. Reduce the size of population using the dynamic reduction
38.  Use Eq. 22 to compute $NP(g+1)$ for next generation
39.    Compute the number of deleted individuals $NP_{diff} = NP(g) - NP(g+1)$
40.    Sort the population $P_g$ based on the individuals' fitness values
41.  Delete the $NP_{diff}$ from $P_g$ which have the worst ranking
42. EndWhile
```

Fig. 2. A Pseduo-code of iDE*a*Sm algorithm

5. Experimental Results and Analysis

The performance of iDE*a*Sm is tested using two benchmark sets: bound-constrained benchmark problems and real-world optimization problems. Also, two well-known engineering design problems are used. A state-of-the-art-algorithms and other evolutionary algorithms are used for comparison. The numerical results and comparisons for the two benchmark sets are presented in the following sub-sections.

5.1. Experiment 1: Bound-constrained Benchmark

In this experiment, 15 bound-constrained problems are tested on 10*D* and 30*D*. Table A.1 in the appendix presents a description of those functions. iDE*a*Sm was coded using Matlab 2017*a*. The following state-of-the-art algorithms are used in our comparisons: JADE [50], CoDE [46], SaDE [31], EPSDE [29], ESMDE [33] and L-SHADE [42]. The same parametric setting is used for these algorithms as reported from their respective sources. The algorithms used 10,000x*D* as the maximum number of function evaluations, and 30 independent runs are used for each problem. In iDE*a*Sm, the same parametric settings for L-SHADE are used in which the initial values for μF and μCR are both set to 0.5 with a memory size equals to 6. $NP_{\min}$ and $NP_{\max}$ are set to 4 and $18 \times D$, respectively for dynamic reduction. For the proposed surrogate model, extensive testing revealed that the best number of chances (*w*) is set to 10.

5.1.1. Discussion of Results: Low Dimensional Problems

The statistical results (mean and standard deviation) of the best final function value for all the algorithms, i.e., $f(\vec{X}_{best})$, *D*=10 and *D*=30 are presented in Tables 1 and 2, respectively. To judge whether there is a statistical significance between iDE*a*Sm and every other contestant algorithm, the Wilcoxon test is used [38] with a 0.05 significance level. In these tables, a "+" sign identifies the problems where iDE*a*Sm algorithm exhibits an inferior performance, "=" sign denotes a tie and so a not statistically significant difference and a "–" sign signifies the problem when the compared algorithm fails against the proposed algorithm and hence iDE*a*Sm exhibits a superior performance. The last row of the results tables summarizes the results of Wilcoxon as *w*(win) / *t*(tie) / *l*(lose). Figs. 3 and 4 show the convergence graphs on 10*D* and 30*D* for some selected functions: (f_2, f_4, f_7, f_{10}, f_{13}), that cover all functions categories.

The results in Table 1 show that the proposed algorithm is at least as good as the other compared algorithms in all the functions. For functions: (f_1, f_8, f_9, f_{10}, f_{12}, and f_{13}) iDE*a*Sm shows superior performance and hence the effectiveness of using the proposed adapted surrogate model to assist the evolutionary search to find better solutions. The algorithms have the same mean values for functions (f_3, f_5, f_6, f_{11}, f_{14}, and f_{15}). Comparing iDE*a*Sm with L-SHADE algorithm shows that iDE*a*Sm has advanced performance and this proves that the proposed surrogate model with *theta* adaptation has the ability to mimic the original objective function. Hence, it is able to generate a competitive offspring at each generation. L-SHADE algorithm shows inferior performance in 7 functions, equal performance in 8 functions and does not win in any of the functions. The convergence graphs in Fig 3 show that iDE*a*Sm has a better convergence rate when compared to the other contestant algorithms. This is due to the use of the proposed surrogate model which helps to accelerate the performance to generate good solutions at each generation and hence improve the convergence.

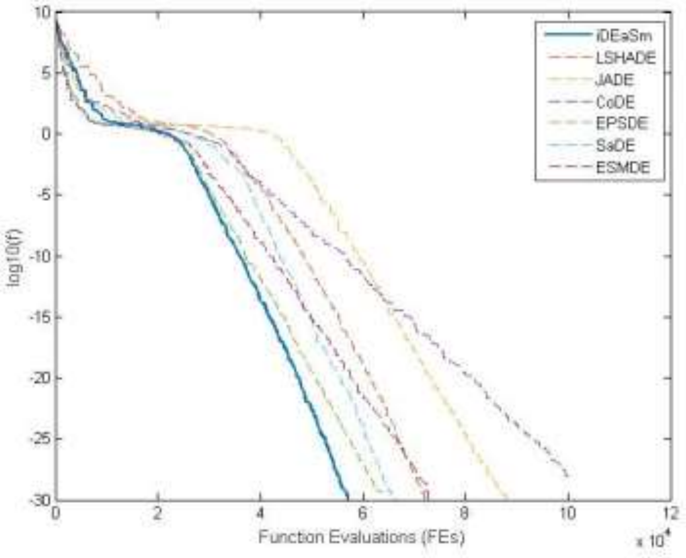
The performance of the proposed algorithm has not substantially decreased at 30*D* as shown in Table 2. iDE*a*Sm was at least as good as other algorithms except for f_{12} in which EPSDE algorithm has better mean value (1.2800E-62). For functions: (f_3, f_{14} and f_{15},), the compared algorithms have the same mean values. Comparing to L-SHADE algorithm, the proposed algorithm wins in 7 functions and has equal performance in 8 functions. As in 10*D*, L-SHADE algorithm does not win in any of the functions. This is evident that using our proposed surrogate model enhances the performance of L-SHADE to generate better solutions. The convergence graphs in Fig. 4 show that iDE*a*Sm has a better convergence rate compared to other compared algorithms.

Table 1. The statistical results of the best function values (mean and standard deviation) @ 10*D*. Best entries are marked in boldface**.**

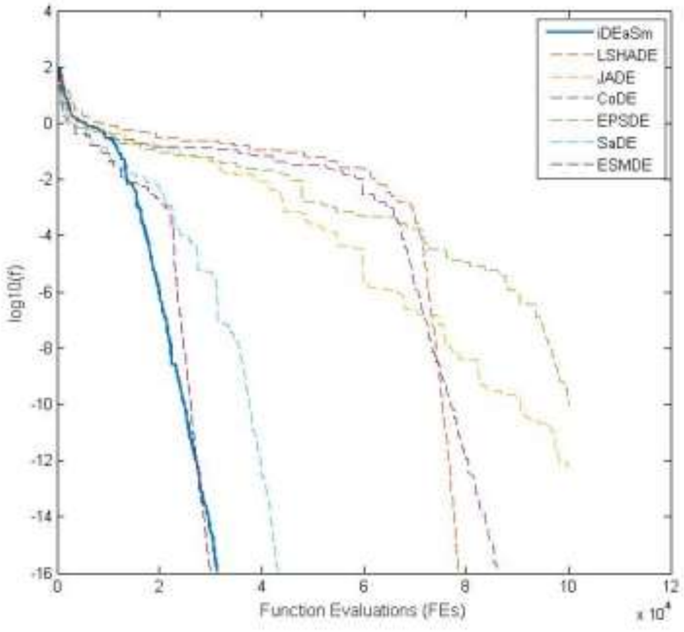
	JADE [50]	CoDE [46]	SaDE [31]	EPSDE [29]	ESMDE [27]	L-SHADE [42]	iDE*a*Sm
f_1	1.1500E-43 (3.2600E-43) –	1.4400E-58 (4.5200E-58) –	2.4800E-101 (1.2000E-100) –	1.6800E-98 (4.5100E-98) –	2.9852E-80 (2.3327E-80) –	3.7870E-114 (2.0459E-113) –	**2.9475E-170** **(0.0000E+00)**
f_2	1.2600E+00 (2.1400E+00) –	1.7300E-26 (4.6700E-26) –	1.33E-01 (7.2800E-15) –	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)**
f_3	**3.5500E-15** **(0.0000E+00)** =	**3.4300E-15** **(6.4900E-16)** =	**1.3000E-15** **(1.7400E-15)** =	**1.4200E-15** **(1.7700E-15)** =	**3.5527E-15** **(0.0000E+00)** =	**3.3159E-15** **(9.0135E-16)** =	**3.0790E-15** **(1.2283E-15)**
f_4	5.5000E-12 (1.2100E-11) –	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	2.5400E-04 (1.3500E-03) –	**0.0000E+00** **(0.0000E+00)** =	7.4528E-02 (1.7456E-02) –	**0.0000E+00** **(0.0000E+00)**
f_5	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)**
f_6	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)**
f_7	3.9500E+00 (2.1600E+01) –	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	7.9000E+00 (3.0100E+01) –	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)**
f_8	2.8500E-59 (1.4800E-58) –	1.6300E-34 (3.5400E-34) –	8.7200E-77 (4.7700E-76) –	3.3700E-74 (4.8800E-74) –	6.5437E-38 (1.2576E-37) –	5.3009E-85 (1.7796E-84) –	**1.3879E-118** **(4.1995E-118)**
f_9	1.3000E-53 (2.3000E-53) –	3.4000E-30 (1.0400E-29) –	1.1500E-64 (4.7400E-64) –	1.0500E-65 (3.0500E-65) –	1.1369E-12 (8.1414E-13) –	4.1457E-67 (2.2546E-66) –	**1.6758E-85** **(6.6526E-85)**
f_{10}	7.0700E-39 (1.9600E-38) –	3.9400E-56 (5.2700E-56) –	5.6000E-99 (1.5000E-98) –	2.2100E-95 (5.1100E-95) –	2.3384E-76 (2.0280E-76) –	3.0307E-110 (1.5482E-109) –	**8.8838E-167** **(0.0000E+00)**
f_{11}	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)** =	**0.0000E+00** **(0.0000E+00)**
f_{12}	3.6800E-22 (6.4800E-22) –	1.4700E-31 (1.1200E-31) –	4.0300E-52 (6.8400E-52) –	4.1300E-49 (8.9400E-49) –	2.1398E-40 (1.0714E-40) –	1.9355E-56 (3.1379E-56) –	**1.0719E-78** **(5.5385E-78)**
f_{13}	2.8800E-24 (4.9000E-24) –	5.3300E-17 (6.5300E-17) –	3.87E-27 (7.5200E-27) –	7.9300E-22 (6.9000E-22) –	4.6566E-18 (2.3130E-18) –	8.2963E-41 (2.7014E-40) –	**1.0887E-57** **(3.2918E-57)** **–**
f_{14}	**1.5700E-32** **(5.5700E-48)** =	**1.5700E-32** **(5.5700E-48)** =	**1.5700E-32** **(5.5700E-48)** =	**1.5700E-32** **(5.5700E-48)** =	**1.5705E-32** **(5.5674E-48)** =	**1.5705E-32** **(5.5674E-48)** =	**1.5705E-32** **(5.5674E-48)**
f_{15}	**1.3500E-32** **(5.5700E-48)** =	**1.3500E-32** **(5.5700E-48)** =	**1.3500E-32** **(5.5700E-48)** =	**1.3500E-32** **(5.5700E-48)** =	**1.3498E-32** **(5.5674E-48)** =	**1.3498E-32** **(5.5674E-48)** =	**1.3498E-32** **(5.5674E-48)**
w/t/l	*0/6/9*	*1/6/8*	*1/6/8*	*0/7/8*	*0/8/7*	*0/8/7*	

Table 2. The statistical results of the best function values (mean and standard deviation) @ 30*D*. Best entries are marked in boldface**.**

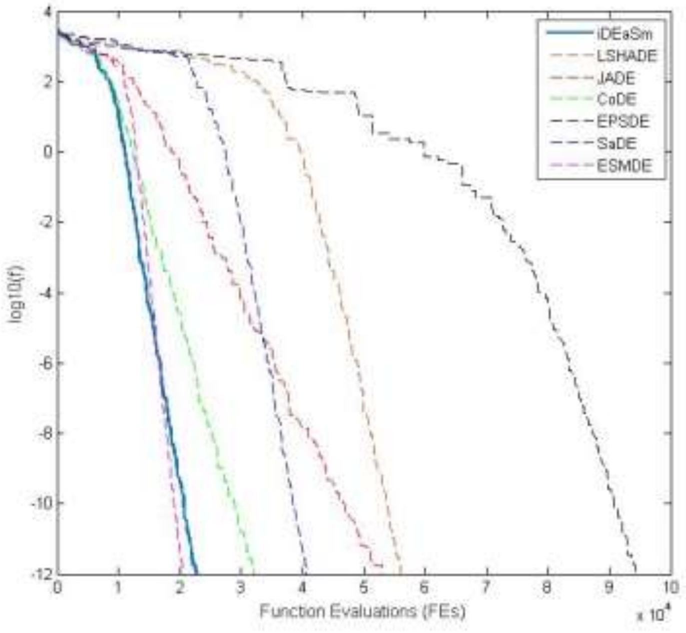
	JADE [50]	CoDE [46]	SaDE [31]	EPSDE [29]	ESMDE [27]	L-SHADE [42]	iDE*a*Sm
f_1	1.2127E-22 (1.2367E-22) −	7.2000E-61 (1.7200E-60) −	2.8646E-87 (3.1584E-87) −	4.2600E-122 (7.6100E-122) =	1.3254E-88 (9.2662E-89) −	3.0515E-94 (6.3433E-94) −	**6.7602E-121 (1.3266E-119)**
f_2	5.3100E-01 (1.3800E+00) −	1.1500E-07 (3.8900E-07) −	9.3000E-01 (1.7200E+00) −	3.9900E-01 (1.2200E+00) −	1.6713E-01 (1.1763E-01) −	1.5062E-24 (1.3560E-25) −	**3.5206E-26 (1.7179E-25)**
f_3	**3.5500E-15 (0.0000E+00)** =	**3.5500E-15 (0.0000E+00)** =	**3.5500E-15 (0.0000E+00)** =	**3.5500E-15 (0.0000E+00)** =	**4.8554E-15 (1.7413E-15)** =	**3.5527E-15 (0.0000E+00)** =	**3.5527E-15 (0.0000E+00)**
f_4	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	**3.5300E-03 (7.0100E-03)** −	**9.0400E-04 (2.7800E-03)** −	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)**
f_5	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	1.1250E-15 (1.3587E-15) −	**0.0000E+00 (0.0000E+00)**
f_6	**0.0000E+00 (0.0000E+00)** =	**3.4500E-05 (1.0600E-04)** −	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)**
f_7	3.9500E+00 (2.1600E+01) −	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	1.5900E+02 (1.4100E+02) −	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)**
f_8	7.9961E-22 (1.1517E-21) −	1.8900E-16 (2.8100E-16) −	6.8500E-22 (2.9200E-21) −	2.8800E-29 (1.1500E-28) −	9.9716E-01 (2.8631E-01) −	**2.0052E-43 (4.9213E-43)** =	**3.9952E-43 (2.9215E-42)**
f_9	3.0500E-14 (1.5900E-13) −	1.0600E-03 (2.3100E-03) −	6.6800E-01 (1.2200E+00) −	1.1900E-08 (3.2900E-08) −	7.8741E+01 (2.3448E+01) −	**2.7561E-26 (8.4671E-25)** =	**9.4197E-26 (2.6766E-25)**
f_{10}	8.3900E-108 (4.6000E-107) −	3.9100E-58 (8.7300E-58) −	3.4801E-84 (3.2820E-84) −	**9.4100E-118 (4.7000E-117)** =	2.0878E-84 (1.4929E-84) −	3.9670E-89 (1.3037E-88) −	**2.0254E-114 (3.5294E-115)**
f_{11}	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	4.1300E-03 (1.4700E-02) −	**0.0000E+00 (0.0000E+00)** −	**0.0000E+00 (0.0000E+00)** −	**0.0000E+00 (0.0000E+00)** −	**0.0000E+00 (0.0000E+00)**
f_{12}	4.3200E-31 (1.5800E-30) −	4.9300E-31 (6.9000E-31) −	4.1505E-52 (2.1042E-52) =	**1.2800E-62 (1.3200E-62)** +	4.2896E-44 (1.7398E-44) −	1.3151E-43 (6.7087E-43) −	1.1997E-52 (1.6926E-52)
f_{13}	2.1237E-09 (1.2216E-09) −	5.4700E-11 (3.7500E-11) −	4.4200E-23 (1.5100E-22) −	5.7400E-13 (4.6200E-13) −	4.8086E-08 (1.5683E-08) −	2.0321E-23 (4.1249E-23) −	**8.1536E-26 (1.7526E-25)**
f_{14}	**1.5700E-32 (5.5700E-48)** =	**1.5700E-32 (5.5700E-48)** =	**1.5700E-32 (5.5800E-48)** =	**1.5700E-32 (5.5700E-48)** =	**1.5705E-32 (5.5674E-48)** =	**1.5705E-32 (5.5674E-48)** =	**1.5705E-32 (5.5674E-48)**
f_{15}	**1.3500E-32 (5.5700E-48)** =	**1.3500E-32 (5.5700E-48)** =	**7.3330E-04 (2.7900E-03)** −	**1.3500E-32 (5.5700E-48)** =	**1.3498E-32 (5.5674E-48)** =	**1.3498E-32 (5.5674E-48)** =	**1.3498E-32 (5.5674E-48)**
w/t/l	0/7/8	0/7/8	0/5/11	1/7/8	0/6/9	0/8/7	


iDEaSm
LSHADE
JADE
CoDE
EPSDE
SaDE
ESMDE
log10(f)
Function Evaluations (FEs)
x 10^4

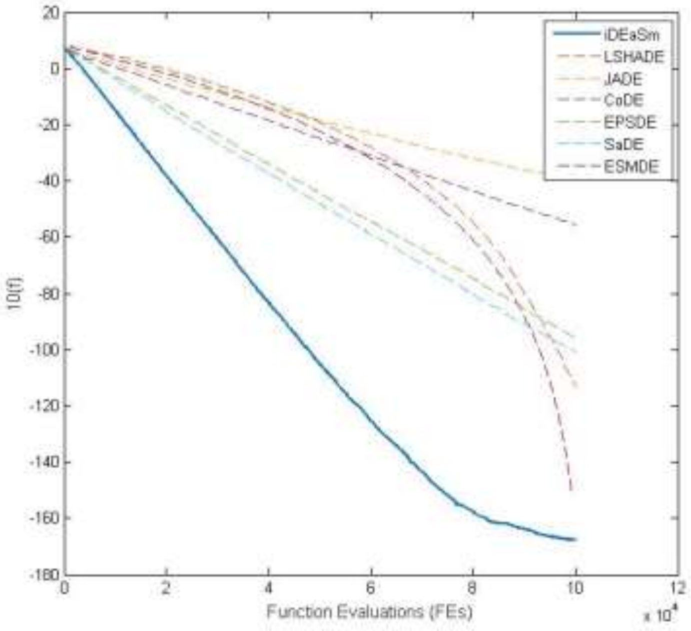
(a)


iDEaSm
LSHADE
JADE
CoDE
EPSDE
SaDE
ESMDE
log10(f)
Function Evaluations (FEs)
x 10^4

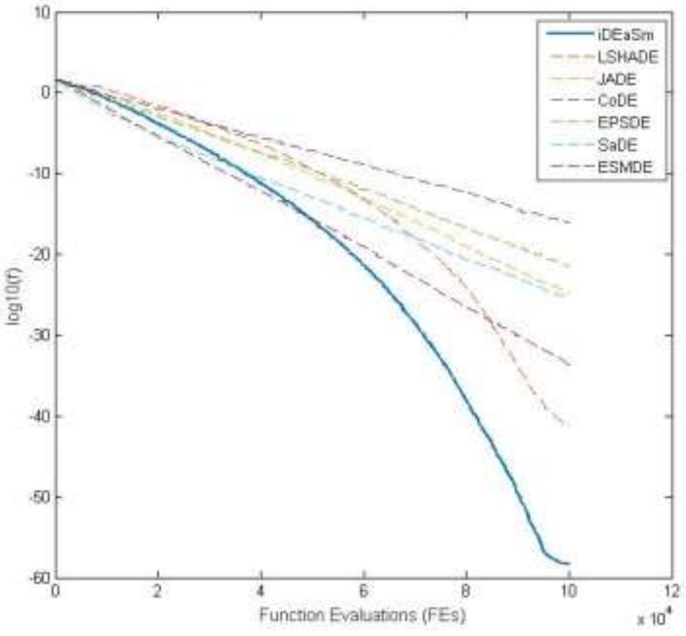
(b)


iDEaSm
LSHADE
JADE
CoDE
EPSDE
SaDE
ESMDE
log10(f)
Function Evaluations (FEs)
x 10^4

(c)

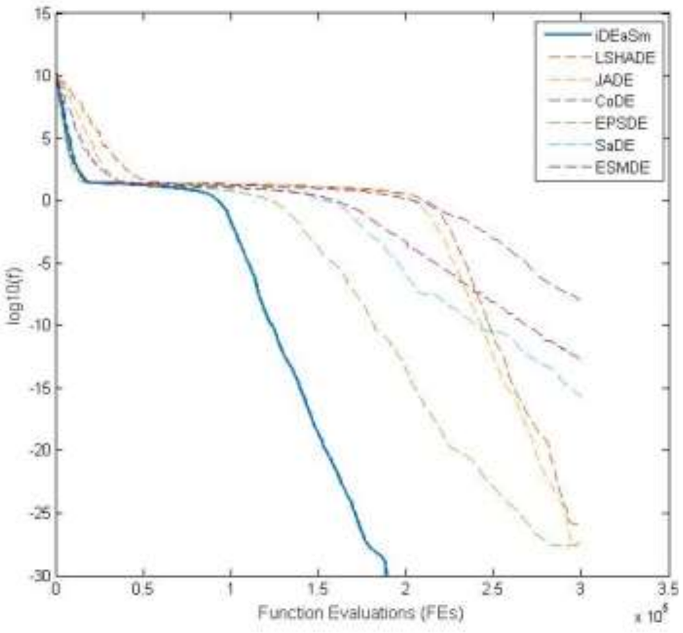

iDEaSm
LSHADE
JADE
CoDE
EPSDE
SaDE
ESMDE
10(f)
Function Evaluations (FEs)
x 10^4

(d)

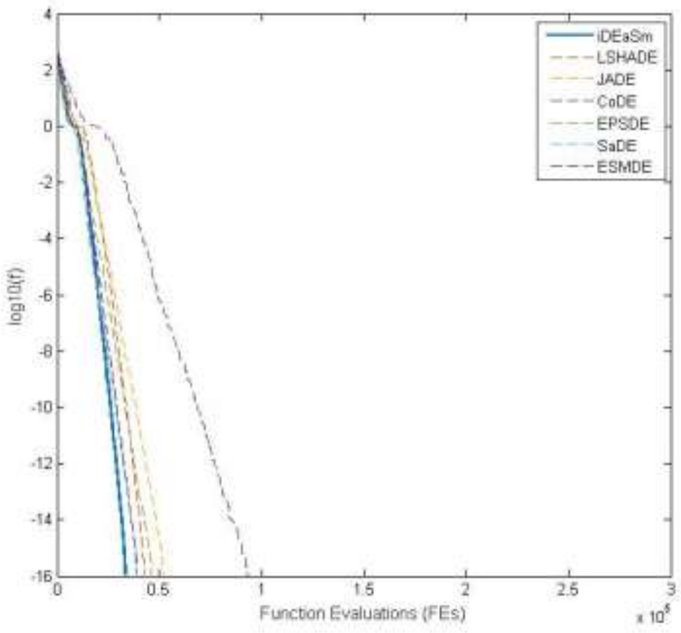


(e)

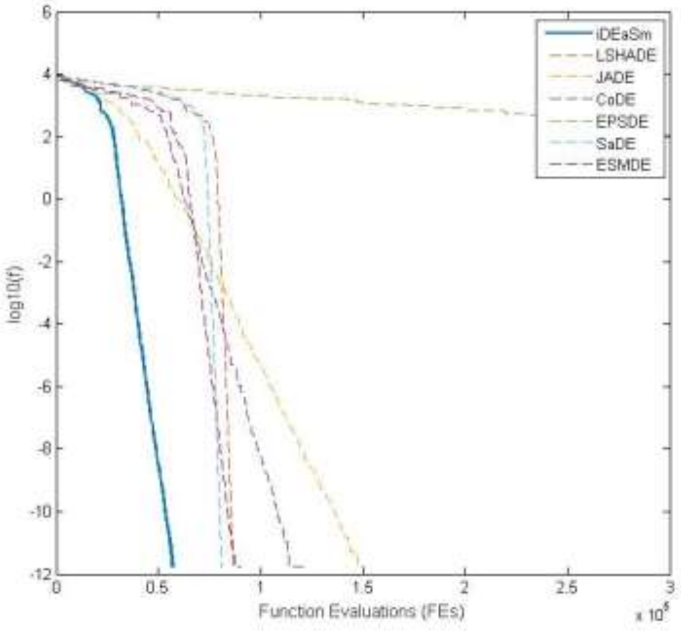
Fig. 3. The convergence graph at 10D, for all the algorithms on some selected functions ((a) f_2, (b) f_4, (c) f_7, (d) f_{10}, (e) f_{13}).



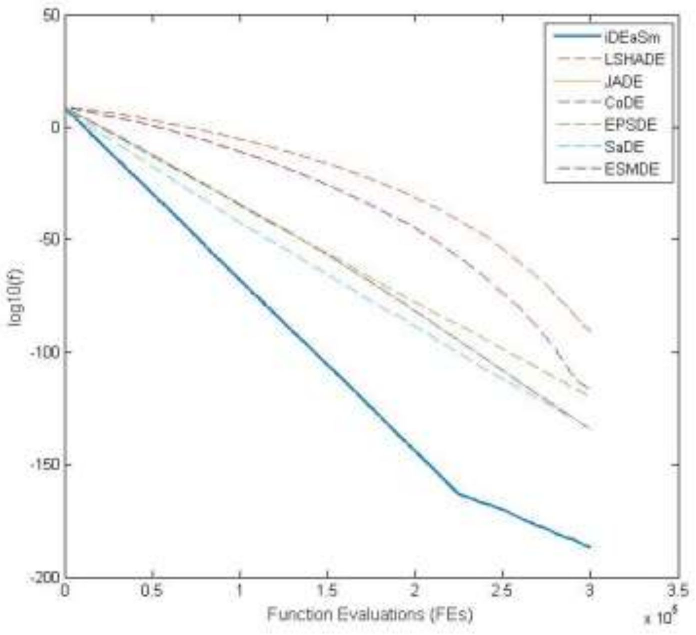
(a)



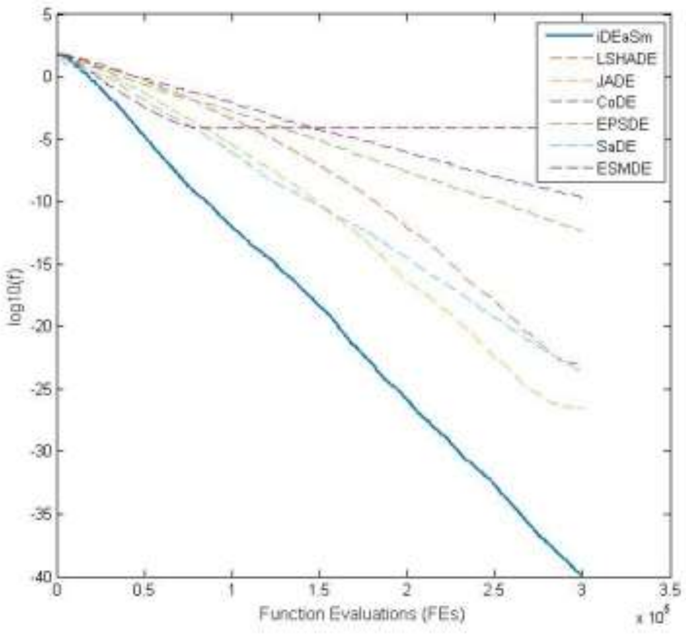
(b)



(c)



(d)



(e)

Fig. 4. The convergence graph at 30*D*, for all the algorithms on some selected functions ((a) f_2, (b) f_4, (c) f_7, (d) f_{10}, (e) f_{13}).

Furthermore, each of the competitor algorithm is ranked using the average Friedman ranking [10]. This test is a non-parametric multiple comparison which takes an input of *n* algorithms and perform a statistical analysis to rank each algorithm. The analysis highlights the differences of the performances which are statistical significant of at least one pair of the compared algorithms. Table 3 present the average Friedman ranking of the competitor algorithms on 10*D* and 30*D*. The test-statistic for this test and the corresponding measured *p*-value are shown in the last two rows of Table 3. The lowest rank is assigned to the best performing algorithm. The results reveal that the proposed algorithm is robust and able to obtain the best rank which is equal to 2.4 and 2.0 for 10*D* and 30*D* respectively. The second performing algorithm is L-SHADE with a rank equals to 3.2 and 3.4 with a clear difference comparing to iDE*a*Sm on both 10*D* and 30*D* respectively.

Table 3
Average Ranking of competitor algorithms, achieved by the Friedman test at dimensions *D*=10 and *D*=30. The best entry is marked in bold.

Algorithm	Ranking (*D*=10)	Ranking (*D*=30)
JADE [50]	5.2667	4.3667
CoDE [46]	4.9000	4.6000
SaDE [31]	3.4333	4.6667
EPSDE [29]	4.1000	3.5000
ESMDE [27]	4.6000	4.6333
L-SHADE [42]	3.2333	3.4333
iDE*a*Sm	**2.4667**	**2.8000**
Statistic	19.42857	10.77143
p-value	0.003498	0.095703

5.1.2. Algorithm Complexity

The complexity of the iDE*a*Sm is shown in Table 4. The algorithm complexity is estimated by $(\hat{T}_2 - T_1)/T_0$, where T_1 denotes the computing time of function f_3 for 2×10^5 evaluations of a certain dimension *D*. The complete computing time for iDE*a*Sm with 2×10^5 evaluations at the same function f_3 with dimension *D* is called T_2. The average of five obtained T_2 values of the algorithm is called $\hat{T}_2$. T_0 is defined as the computing time of the following code:

```
for i=1:1000000
    x = 0.55+(double)i; x = x+x; x = x/2; x=x*x; x=sqrt(x);
    x=log(x); x=exp(x); x=x/(x+2);
end
```

The complexity of the algorithm is compared to the other compared algorithms, in Table 5. The extra time that the proposed algorithm takes compared to L-SHADE algorithm is for building the surrogate model.

Table 4. Algorithm Complexity

Dimension	T_0	T_1	$\hat{T}_2$	$(\hat{T}_2 - T_1)/T_0$
D=10	1.2970E-01	3.0071E+00	1.0215E+01	5.5572E+01
D=30	1.2970E-01	1.0873E+01	2.3146E+01	9.4623E+01

Table 5. A Comparison of Algorithms Complexities

	Value of $(\hat{T}_2 - T_1)/T_0$ at the different dimensions	
Algorithm	*D*=10	*D*=30
iDE*a*Sm	5.5572E+01	9.4623E+01
LSHADE [42]	3.1762E+01	7.4135E+01
JADE [50]	3.4513E+01	7.4531E+01
CoDE [46]	4.4114E+01	9.3725E+01
EPSDE [29]	5.5600E+01	1.5200E+02
SaDE [31]	4.1658E+01	1.3185E+02
ESMDE [27]	7.7937E+01	2.5350E+02

5.1.3. Scalability Test

To test the ability of iDE*a*Sm to solve the benchmark problems with higher dimensions, we used to increase the dimensionality to 50 and 100. The results are presented in Tables 6 and 7 for 50*D* and 100*D* respectively. The results include the mean and standard deviation, and the Wilcoxon is also used to check the significance of the results for each test problem compared to iDE*a*Sm algorithm. The results show that the proposed algorithm is able to solve the 50*D* problems efficiently. When compared to L-SHADE algorithm, iDE*a*Sm shows a better and equal performances in 14 functions and an equal performance in only one function which is f_{11}. The ESMDE algorithm is better than iDE*a*Sm in 2 functions (f_{11} and f_{12}) and shows an inferior performances in 6 functions. For the other algorithms: JADE, CoDE, SaDE and EPSDE, the number of functions in which those algorithms show an inferior performance is within the range 8-13. For 100D, the results presented in Table S.3 show that iDE*a*Sm algorithm was able to provide competitive performance when compared to other algorithms. The EPSDE algorithm was able to win in two functions (f_2 and f_{12}) and fails in 12 functions when it compared to iDE*a*Sm algorithm. Comparing L-SHADE with iDE*a*Sm algorithm, it shows a superior performance in f_{13}, an equal performance in 6 functions and an inferior performance in 8 functions. The average Friedman ranking for the compared algorithm is also presented in Table 8. The proposed algorithm has the lowest/best rank which equals to 2.3 and 1.83 for both 50D and 100D compared to other algorithms. The results affirm the efficiency of using the proposed surrogate model to solve the benchmark problems for the different dimensions.

Table 6. The statistical results of the best function values (mean and standard deviation) @ 50*D*. Best entries are marked in boldface**.**

	JADE [50]	CoDE [46]	SaDE [31]	EPSDE [29]	ESMDE [27]	L-SHADE [42]	iDE*a*Sm
f_1	1.5666E-60 (1.7516E-60) −	1.7537E-50 (2.8213E-46) −	2.8184E-71 (6.9942E-70) −	**3.3476E-100 (6.1469E-100)** =	1.8048E-87 (8.0831E-88) −	9.5166E-82 (2.9309E-81) −	**2.0686E-96 (5.4766E-96)**
f_2	9.3021E-01 (1.7150E+00) −	1.0749E+01 (1.5846E+01) −	1.4010E+00 (2.2982E+00) −	1.0631E+00 (1.7931E+00) −	1.1892E+01 (8.2394E-01) −	**2.0112E-04 (6.7815E-04)** =	**1.3784E-04 (2.0318E-04)**
f_3	**6.7502E-15 (1.0840E-15)** =	**3.5527E-15 (0.0000E+00)** =	4.5635E-01 (5.8706E-01) −	3.2623E-01 (4.7359E-01) −	**6.6317E-15 (1.2283E-15)** =	**6.2765E-15 (1.5283E-15)** =	**5.3291E-15 (1.8067E-15)**
f_4	4.9307E-04 (1.8764E-03) −	**0.0000E+00 (0.0000E+00)** =	5.0855E-03 (1.0538E-02) −	3.6911E-03 (7.6335E-03) −	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)**
f_5	**0.0000E+00 (0.0000E+00)** =	4.2885E-10 (1.1700E-09) −	4.9748E-01 (8.9565E-01) −	**0.0000E+00 (0.0000E+00)** =	**0.0000E+00 (0.0000E+00)** =	4.2006E-11 (5.1357E-11) −	**0.0000E+00 (0.0000E+00)**
f_6	**0.0000E+00 (0.0000E+00)** =	3.5927E+01 (2.4043E+00) −	6.0000E-01 (6.7466E-01) −	6.2689E-05 (1.7105E-04) −	**0.0000E+00 (0.0000E+00)** =	9.0955E-11 (1.3131E-10) −	**0.0000E+00 (0.0000E+00)**
f_7	3.9479E+00 (2.1624E+01) −	**1.8190E-11 (0.0000E+00)** =	**1.8190E-11 (0.0000E+00)** =	5.8664E+02 (2.1986E+02) −	**1.8190E-11 (0.0000E+00)** =	**4.0745E-11 (2.3194E-11)** =	**3.1044E-11 (7.3839E-12)**
f_8	2.0788E-10 (3.4582E-10) −	1.3043E-09 (2.1077E-09) −	5.3968E-12 (7.6056E-12) −	1.1743E-16 (1.4536E-16) −	5.2356E+00 (1.7463E+00) −	4.2887E-15 (5.1659E-15) −	**8.5637E-21 (2.4961E-20)**
f_9	7.3951E-02 (1.0589E-01) −	7.5198E+01 (7.2515E+01) −	1.4678E+02 (8.4943E+01) −	1.9235E+01 (3.6545E+01) −	4.4176E+02 (9.6750E+01) −	**1.5519E-10 (4.1843E-10)** =	**1.6958E-08 (3.1703E-08)**
f_{10}	2.3721E-80 (2.5716E-80) −	1.4696E-50 (4.9629E-50) −	1.3356E-75 (6.2743E-74) −	**2.2544E-90 (6.6456E-90)** =	2.8777E-83 (1.3322E-83) −	3.0384E-75 (7.5218E-75) −	**4.8147E-89 (1.4048E-88)**
f_{11}	1.1554E-02 (4.6505E-02) =	**0.0000E+00 (0.0000E+00)** +	1.9047E-01 (2.5950E-01) −	6.0935E-02 (1.5417E-01) =	**0.0000E+00 (0.0000E+00)** +	4.7070E-07 (2.5781E-06) +	6.3577E-03 (2.2292E-02)
f_{12}	1.2500E-28 (6.8454E-27) −	1.2364E-36 (1.0964E-36) =	5.2210E-30 (5.7970E-30) −	**3.6586E-68 (8.2291E-68)** +	6.6203E-44 (1.6691E-44) +	1.7289E-31 (3.6552E-31) −	3.8858E-38 (1.3880E-37)
f_{13}	7.6558E-06 (1.4430E-06) −	9.0302E-10 (1.6608E-09) −	2.4171E-11 (1.2602E-10) −	3.3499E-05 (9.1893E-05) −	4.1276E-04 (1.3855E-04) −	3.6982E-11 (4.8581E-11) −	**4.4949E-14 (5.0427E-14)**
f_{14}	3.4556E-03 (1.8927E-02) −	**1.5705E-32 (5.5674E-48)** =	**1.5878E-32 (9.4265E-34)** =	6.9106E-03 (3.7851E-02) −	**1.5705E-32 (5.5674E-48)** =	**1.5705E-32 (5.5674E-48)** =	**1.5705E-32 (5.5674E-48)**
f_{15}	**1.3498E-32 (5.5674E-48)** =	**1.3498E-32 (5.5674E-48)** =	7.6907E-03 (3.2018E-02) −	4.7611E-03 (1.8176E-02) −	**1.3498E-32 (5.5674E-48)** =	**1.3498E-32 (5.5674E-48)** =	**1.3498E-32 (5.5674E-48)**
w/t/l	*0/5/10*	*1/6/8*	*0/2/13*	*1/4/10*	*2/7/6*	*1/7/7*	

Table 7. The statistical results of the best function values (mean and standard deviation) @ 100*D*. Best entries are marked in boldface**.***

	JADE [50]	CoDE [46]	SaDE [31]	EPSDE [29]	L-SHADE [42]	iDE*a*Sm
f$_1$	1.5842E-06 (4.3826E-05) −	1.2865E-36 (1.7681E-35) −	1.0149E-40 (3.9873E-40) −	2.9261E-60 (8.2926E-60) −	4.3809E-62 (7.3436E-62) −	1.9438E-68 (1.0600E-68)
f$_2$	3.9866E-01 (1.2164E+00) +	1.0776E+02 (4.6948E+01) −	6.4241E+01 (5.6063E+01) −	3.4871E+01 (3.7129E+01) +	4.7417E+01 (1.8527E+00) =	4.6938E+01 (2.0415E+00)
f$_3$	9.9685E-01 (4.9906E-01) −	3.5527E-15 (0.0000E+00) =	2.5378E+00 (5.0070E-01) −	2.4028E+00 (3.8566E-01) −	9.2371E-15 (3.1776E-15) =	7.1054E-15 (5.4713E-15)
f$_4$	2.7935E-03 (5.1674E-03) −	9.0369E-04 (2.7825E-03) −	4.3677E-02 (1.1494E-01) −	1.2778E-02 (2.5667E-02) −	0.0000E+00 (0.0000E+00) =	0.0000E+00 (0.0000E+00)
f$_5$	5.9212E-04 (3.2432E-04) =	1.0942E+02 (2.0918E+01) −	1.0182E+01 (2.1014E+00) −	1.8706E-04 (1.0179E-04) =	1.2854E-02 (1.0690E-02) −	1.2233E-03 (7.6385E-04)
f$_6$	9.7316E-03 (7.3416E-03) =	1.7273E+02 (2.3223E+01) −	2.0000E+01 (1.8027E+01) −	1.7936E+01 (3.8830E+01) −	2.9459E-01 (4.3859E-01) −	1.3597E-04 (4.3130E-04)
f$_7$	1.1844E+01 (3.6139E+01) −	3.9479E+00 (2.1624E+01) −	1.6581E+02 (1.3770E+02) −	2.5795E+03 (4.1276E+02) −	1.5771E-02 (6.8273E-03) =	3.2724E-02 (2.2491E-02)
f$_8$	3.2654E-05 (5.1923E-05) −	1.0645E-03 (6.5247E-04) −	1.0246E-03 (1.2768E-03) −	4.9676E-06 (3.4005E-06) −	2.1943E-08 (2.1227E-08) =	2.3920E-08 (1.2017E-08)
f$_9$	1.1558E+03 (4.9123E+02) −	3.1149E+03 (9.8552E+02) −	2.7370E+03 (6.5783E+02) −	3.0339E+03 (1.1518E+03) −	9.8332E-01 (6.1641E-01) −	6.8209E-03 (2.7436E-03)
f$_{10}$	7.8412E-42 (5.7126E-41) −	9.7625E-31 (2.7874E-31) −	1.2517E-36 (5.5244E-36) −	6.8268E-46 (1.6265E-45) −	6.5130E-51 (1.4426E-50) −	6.2817E-55 (6.0264E-55)
f$_{11}$	2.6972E+00 (1.1407E+00) −	5.9385E-02 (1.7925E-01) =	6.5940E+00 (2.4510E+00) −	5.5270E+00 (1.6266E+00) −	2.0067E-01 (1.2162E-01) −	4.8360E-02 (5.7126E-02)
f$_{12}$	1.8774E-15 (9.5304E-15) −	6.8588E-22 (1.4127E-22) =	4.7431E-20 (2.5979E-20) −	1.2923E-27 (5.3976E-26) +	2.0236E-21 (5.4507E-21) −	6.9278E-23 (6.6317E-23)
f$_{13}$	1.2677E+00 (8.2301E-01) −	1.1712E-01 (1.6938E-01) −	9.6197E+00 (6.2407E+00) −	4.1912E+00 (1.5971E+00) −	1.3468E-08 (7.5128E-09) +	8.2334E-06 (3.9161E-06)
f$_{14}$	1.8287E-32 (5.7125E-33) =	1.5705E-32 (5.5674E-48) =	3.1097E-02 (1.0239E-01) −	3.4556E-02 (1.0661E-01) −	1.6738E-32 (2.5001E-33) =	2.0869E-32 (6.3235E-33)
f$_{15}$	1.4650E-03 (8.0239E-03) −	1.0987E-03 (3.3526E-03) −	4.4324E-02 (1.5732E-01) −	1.0987E-03 (3.3526E-03) −	1.3662E-32 (4.2616E-34) =	1.3498E-32 (5.4612E-36)
w/t/l	*1/3/11*	*0/4/11*	*0/0/15*	*2/1/12*	*1/6/8*	

* The ESMDE algorithm has been excluded for 100*D* due to its extra computational time.

Table 8
Average Ranking of competitor algorithms, achieved by the Friedman test at dimensions *D*=50 and *D*=100. The best entry is marked in bold.

Algorithm	Ranking (*D*=50)	Ranking (*D*=100)
JADE [50]	4.5000	3.7333
CoDE [46]	4.3000	4.0333
SaDE [31]	5.4000	5.2000
EPSDE [29]	4.3000	3.8333
ESMDE [27]	3.7333	N.A
L-SHADE [42]	3.4667	2.3667
iDE*a*Sm	**2.3000**	**1.8333**
Statistic	18.11429	31.72380
p-value	0.005953	6.73830E-06

N.A: Not Available

5.2. Experiment 2: Real-world Optimization Benchmark

The performance of iDE*a*Sm is also tested on real-world optimization problems to test the validity of the proposed surrogate model to solve this kind of problems. Table 9 summarizes the problems which have been taken from the IEEE CEC'11 on real-world numerical optimization problems [8]. This benchmark spans different challenging problems from different fields such as: sound waves, radar system, network expansion planning, economic dispatch problems and spacecraft trajectory optimization. The dimensions for the used problems vary as shown in Table 9, and they are within the range 6*D*–40*D*.

Table 9. A summary of the used real-world numerical optimization problems.

	Description	*D*
T1	Parameter Estimation of FM Sound Waves	6
T2	Lennard-Jones Potential	30
T5	Tersoff Potential Function Min. Problem-Si(B)	30
T6	Tersoff Potential Function Min. Problem-Si(C)	30
T7	Spread Spectrum Radar Polly phase Code Design	20
T8	Transmission Network Expansion Planning	7
T10	Circular Antenna Array Design	12
T14	Static Economic Load Dispatch – Instance 2	13
T15	Static Economic Load Dispatch – Instance 3	15
T16	Static Economic Load Dispatch – Instance 4	40
T21	Messenger: Spacecraft Trajectory Optimization	26
T22	Cassini 2: Spacecraft Trajectory Optimization	22

5.2.1. Discussion of Results

The performance of iDE*a*Sm is compared with the following algorithms including the winner of the IEEE CEC'11 benchmark competition. The same parametric set-up is used in our algorithm as in experiment 1. Each algorithm is run for 25 independent runs using 50,000 function evaluations, to match the competition rules as can found in [8].

1. EA-DE-MA: performance of a hybrid EA-DE-memetic algorithm [22].
2. ENSML_DE: ensemble differential evolution algorithm [28].
3. WI_DE: self-adaptive cluster based and weed inspired differential evolution algorithm [16]
4. DE-RHC: Benchmarking a Hybrid DE-RHC Algorithm [23].
5. GA-MPC: GA with a new multi-parent crossover, winner of this competition [13].
6. L-SHADE: Improving the Search Performance of SHADE Using Linear Population Size Reduction [42].

Table 10 shows the statistical results including: best, median, worst, mean and standard deviation for iDE*a*Sm at 50,000 function evaluations. A comparison with the aforementioned algorithms is presented in Table 11 showing the mean and standard deviation for the best objective values for the aforementioned algorithms. The Wilcoxon test is also used to judge the significance of the results compared to iDE*a*Sm algorithm. The last row summarizes the number of functions in which each algorithm shows superior, equal and inferior performances compared iDE*a*Sm according to Wilcoxon test. The results of Table 11 show that iDE*a*Sm is successfully used to solve challenging real-world problems, and the mean values for the best objective solutions are at least as good as other algorithms for all the problems except for problem T6. DE-RHC algorithm got the best results for this problem with a mean value equal to -2.8600E+01. L-SHADE algorithm shows an inferior performance in 9 problems, an equal performance in 3 problems and it never wins in any of the 12 used real-world problems. This clearly proves that the proposed surrogate model advances the evolutionary search to generate good solutions for real-world problems. Compared to the winner of this competition (GA-MPC), the proposed algorithm wins in 9 problems and shows a tie in 3 problems. The GA-MPC algorithm was not able to beat iDE*a*Sm in any of the used problems. Moreover, the simulation results show the ability of the proposed surrogate model with the adaptation scheme to mimic the response of changing real-world optimization problems. The average Friedman ranking is presented in Table 12 for each of the used algorithms. The best rank is obtained by iDE*a*Sm and equals to 2.0 compared to the second rank by L-SHADE algorithm which equals to 3.2. The use of such statistical methods proof the effectiveness and robustness of the proposed algorithm.

Table 10. The simulation results for iDE*a*Sm for each test probelm

	Best	Median	Worst	Mean	Std
T1	0.0000E+00	0.0000E+00	9.5452E-09	7.5494E-10	2.0371E-09
T2	-2.8423E+01	-2.7480E+01	-2.5791E+01	-2.7401E+01	6.4837E-01
T5	-3.6845E+01	-3.5500E+01	-3.3111E+01	-3.5547E+01	1.0027E+00
T6	-2.9166E+01	-2.3006E+01	-1.5088E+01	-2.2454E+01	3.5641E+00
T7	5.0000E-01	7.5763E-01	9.6968E-01	7.6208E-01	1.2079E-01
T8	8.0000E+00	2.2000E+02	2.2000E+02	2.2000E+02	0.0000E+00
T10	-2.1842E+01	-2.1445E+01	-2.1391E+01	-2.1508E+01	1.3122E-01
T14	1.4000E+01	1.8114E+04	1.8198E+04	1.8118E+04	5.0147E+01
T15	1.5000E+01	3.2751E+04	3.2771E+04	3.2752E+04	5.9879E+00
T16	1.6000E+01	1.2610E+05	1.2695E+05	1.2606E+05	5.3807E+02
T21	1.1186E+01	1.5473E+01	1.8088E+01	1.5458E+01	1.7363E+00
T22	1.0224E+01	1.8000E+01	2.0888E+01	1.7737E+01	2.4031E+00

Table 11. The statistical results of the best function values (mean and standard deviation). Best entries are marked in boldface**.**

	EA-DE-MA [22]	ENSML_DE [28]	WI_DE [16]	DE-RHC [23]	GA-MPC [13]	L-SHADE [42]	iDE*a*Sm
T1	7.6023E+00 (6.0206E+00) −	7.0600E+00 (6.0400E+00) −	3.5400E+00 (5.9300E+00) −	1.2600E+01 (6.0900E+00) −	1.2678E+00 (4.3986E+00) −	2.8408E+00 (5.1638E+00) −	**7.5494E-10 (2.0371E-09)**
T2	-2.3020E+01 (3.5638E+00) −	-1.2600E+01 (1.8900E+00) −	-1.8300E+01 (3.7800E+00) −	-2.3900E+01 (1.9500E+00) −	-1.1477E+01 (3.1872E+00) −	-2.1001E+01 (1.7039E+00) −	**-2.7401E+01 (6.4837E-01)**
T5	**-3.5204E+01 (1.5347E+00) =**	-2.3800E+01 (2.0500E+00) −	-3.2000E+01 (3.2600E+00) −	**-3.5200E+01 (1.1100E+00) =**	-2.6377E+01 (6.3227E+00) −	-3.3703E+01 (4.0358E-01) −	**-3.5547E+01 (1.0027E+00)**
T6	2.4833E+01 (3.0506E+00) −	-1.2800E+01 (1.9800E+00) −	-2.6200E+01 (2.7800E+00) =	**-2.8600E+01 (1.4100E+00)** +	-1.7332E+01 (3.6484E+00) −	-2.6875E+01 (2.1270E+00) =	-2.2454E+01 (3.5641E+00)
T7	**5.8474E-01 (7.7935E-02) =**	1.6100E+00 (7.7400E-02) −	1.0000E+00 (1.7300E-01) −	1.3100E+00 (1.1700E-01) −	1.6135E+00 (3.7303E-01) −	1.2778E+00 (5.1890E-02) −	**7.6208E-01 (1.2079E-01)**
T8	**2.2000E+02 (0.0000E+00) =**	**2.2000E+02 (0.0000E+00) =**	**2.2000E+02 (0.0000E+00) =**	**2.2000E+02 (0.0000E+00) =**	**2.2000E+02 (0.0000E+00) =**	**2.2000E+02 (0.0000E+00) =**	**2.2000E+02 (0.0000E+00)**
T10	-1.8949E+01 (3.2443E+00) −	-1.5200E+01 (3.5900E+00) −	**-2.1600E+01 (1.9400E-01) =**	-1.5500E+01 (2.3000E+00) −	**-2.1629E+01 (1.2381E-01) =**	**-2.1462E+01 (8.4907E-02) =**	**-2.1508E+01 (1.3122E-01)**
T14	1.9297E+04 (1.8436E+02) −	1.8200E+04 (7.7300E+01) −	1.8400E+04 (1.0200E+02) −	1.9300E+04 (1.9100E+02) −	1.8386E+04 (1.5209E+02) −	1.8131E+04 (6.0957E+01) −	**1.8118E+04 (5.0147E+01)**
T15	3.3022E+04 (7.1048E+01) −	3.2800E+04 (5.2500E+01) −	3.2800E+04 (6.2300E+01) −	3.3100E+04 (7.5300E+01) −	3.2887E+04 (4.6263E+01) −	3.2765E+04 (4.0029E+01) −	**3.2752E+04 (5.9879E+00)**
T16	3.5790E+05 (9.6664E+03) −	1.3200E+05 (2.8300E+03) −	1.4000E+05 (3.9000E+03) −	1.3900E+05 (3.2600E+03) −	1.4129E+05 (2.4745E+03) −	1.2703E+05 (2.3027E+03) −	**1.2606E+05 (5.3807E+02)**
T21	1.6232E+01 (2.9456E+00) −	2.2600E+01 (1.9100E+00) −	**1.4200E+01 (3.0100E+00) =**	2.0200E+01 (2.3300E+00) −	1.8219E+01 (5.5009E+00) −	1.7658E+01 (2.0562E+00) −	**1.5458E+01 (1.7363E+00)**
T22	1.9708E+01 (4.8234E+00) −	1.9900E+01 (5.9600E+00) −	**1.6600E+01 (4.4200E+00) =**	2.0900E+01 (3.1600E+00) −	**1.6034E+01 (2.9284E+00) =**	1.8981E+01 (3.1157E+00) −	**1.7737E+01 (2.4031E+00)**
w/t/l	0/3/9	0/1/11	0/5/7	1/2/9	0/3/9	0/3/9	

Table 12
Average Ranking of competitor algorithms, achieved by the Friedman test for experiment 2. The best entry is marked in bold.

Algorithm	Ranking
EA-DE-MA [22]	4.5833
ENSMLDE [28]	5.2917
WIDE [16]	3.5417
DE-RHC [23]	4.9167
GA-MPC [13]	4.4167
L-SHADE [42]	3.2500
iDEaSm	**2.0000**
Statistic	20.04464
p-value	0.002719

5.3. Experiment 3: Engineering design optimization problems

In this section, we study the performance of iDE*a*Sm to solve real-world engineering optimization problems. Two cases studies are proposed in which two well-known engineering optimization problems are used: welded beam design and pressure vessel design problems. The following sub-sections presents those case studies and the experimental results of using iDE*a*Sm to solve them with comparative comparisons. As those two engineering problems are constrained problems, a classic penalty method is used to handle the constraints [5]. The method uses a penalty coefficient for each constraint violation, and then calculates the fitness value by adding this factor to the objective function.

5.3.1. Welded Beam Design Problem

The well-known welded beam design problem is used [9]. The structure of this problem is presented in Fig. 5 which consists of a weld that is clamped between two beams: A and B. The optimization task is to find the best dimensions for *b*, *t*, *h* and *l* which are used to carry a load *P* and support and generate a minimum fabrication cost.

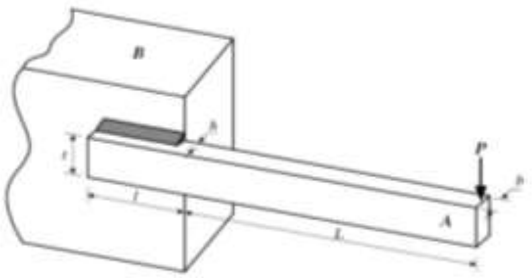


Fig. 5. Framework for the welded beam design problem [9]

The mathematical expression of this problem is expressed as follows which consist of 4 decision variables and 7 constraints to present comprised of the welding labor, material costs, set-up and fabricating cost:

$$
\begin{aligned}
& f(x) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2) \\
& \text{subject to,} \\
& g_1(x) = \tau(x) - \tau_{\max} \le 0, \\
& g_2(x) = \sigma(x) - \sigma_{\max} \le 0, \\
& g_3(x) = x_1 - x_4 \le 0, \\
& g_4(x) = 0.10471 x_1^2 + 0.04811 x_3 x_4 (14.0 + x_2) - 5.0 \le 0, \\
& g_5(x) = 0.125 - x_1 \le 0, \\
& g_6(x) = \delta(x) - \delta_{\max} \le 0, \\
& g_7(x) = P - P_c(x) \le 0
\end{aligned}
\tag{28}
$$

where:

$$\tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2},\ \tau'' = \frac{M\,R}{J},$$

$$M = P(L + \frac{x_2}{2}), R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2},$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})^2\right]\right\},$$

$$\sigma(X) = \frac{6PL}{x_4x_3^2},\ \delta(X) = \frac{4PL^3}{Ex_3^3x_4},$$

$$P_c(X) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}),$$

$P = 6,000\,\text{lb},\ L = 14\,\text{in},\ E = 30\times10^6\ \text{psi},\ G = 12\times10^6\ \text{psi}$

$\tau_{\max} = 30{,}600\ \text{psi},\ \sigma_{\max} = 30{,}000,\ \delta_{\max} = 0.25\,\text{in.}$

Many algorithms have been used to solve this problem including evolutionary algorithms. Table 13 presents a comparative comparison with the best algorithms from literature. In this table, the optimal design variables with the best found function cost are presented for the different methods. Also, the statistical results including best, worst, mean and standard deviation with the number of function evaluation (FEs) which is used to get these values. The results in this table show that iDE*a*Sm is able to generate a similar/better solution among the used methods with the minimum number of function evaluations. The proposed algorithm used 4,425 function evaluations to get a function cost equals to 1.7248. The author in [35] was able to generate the same function cost but with larger number of function evaluations which is equals to 50,000. The results reveal the ability of using iDE*a*Sm to solve this problem with high quality solutions compared to other best reported results for other methods.

Table 13. Comparison of the best solutions found for the welded beam design problem

Methods	Optimal design variables								
	x_1	x_2	x_3	x_4	Best	Worst	Mean	Std	FEs
Kaveh & Talatahari [19]	0.207301	3.435699	9.041934	0.205714	1.723377	1.762567	1.743454	0.007356	4,000
Deb [9]	0.248900	6.173000	8.178900	0.253300	2.433116	N.A	N.A	N.A	N.A
Tomassetti [43]	0.205729	3.470489	9. 036624	0.205730	1.7248	N.A	N.A	N.A	10,000
Aragon et al. [1]	0.244369	6.218613	8.291474	0.244369	2.38113	2.710406	2.439811	0.093146	320,000
Coello &Montes [6]	0.205986	3.471328	9.020224	0.206480	1.728226	1.792654	1.993408	0.074713	N.A
Salimi [35]	0.2057296	3.470488	9.0366239	0.205729	1.7248523	1.7248523	1.7248523	7.7087E-16	50,000
iDE*a*Sm	0.20572963	3.4704888	9.0366238	0.20572965	1.7248523	1.7248864	1.7248564	7.6933E-06	4,425

* N.A: Not Available

5.3.2. Pressure Vessel Design Problem

The pressure vessel problem is a structural engineering problems consists of a compressed air storage tank with a maximum volume of 75ft^3 and a working pressure of 2000psi [36]. As shown in Fig. 6, the hemispherical heads are capped a cylindrical vessel at both ends. The shell is made in two halves using rolled steel plate and they form a cylinder by joining two longitudinal welds. The objective function is to minimize the total cost which consists of cost of forming, welding and material [36].

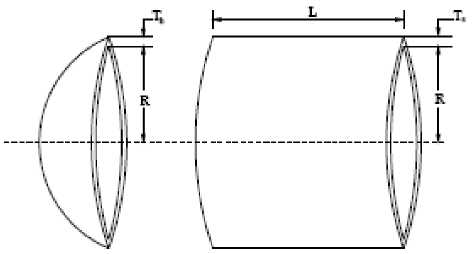


Fig. 6. Design of pressure vessel problem [36].

Four design variables are associated with this problem: thickness of pressure vessel $T_s\,(=x_1)$) thickness of head $T_h\,(=x_2)$, inner radius of the vessel $R(=x_3)$ and length of the vessel without the heads $L(=x_4)$. The mathematical representation of this problem can be expressed as:

$$\begin{aligned}
&\text{Minimize } f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1x_4 + 19.84x_1^2x_3,\\
&\text{subject to,}\\
&g_1(x) = -x_1 + 0.0193x_3 \le 0\\
&g_2(x) = -x_2 + 0.00954x_3 \le 0\\
&g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0\\
&g_4(x) = x_4 - 240 \le 0\\
&\text{where: } 1 \le x_1 \le 0.0625, 1 \le x_2 \le 99, 10 \le x_{3,4} \le 200
\end{aligned} \tag{29}$$

In Table 14, a comparison of the best found solutions for pressure vessel are presented. The results include the optimal design variables, statistical results and the number of FEs to get these results for the cost function. The best function cost is obtained by iDE*a*Sm algorithm which equals to 5988.02732 with 20,000 FEs. The second performing algorithm is the one proposed by [15] and [36] with a cost equals to 6059.71433. The results proof the effectiveness of using the proposed surrogate model to solve such design problem.

Table 14. Comparison of the best solutions found for the pressure vessel design problem

Methods	Optimal design variables								
	x_1	x_2	x_3	x_4	**Best**	**Worst**	**Mean**	**Std**	**FEs**
Sandgren [36]	1.1250	0.6250	48.3807	11.7449	8048.6190	N.A	N.A	N.A	N.A
Huang et. al, [17]	0.8125	0.4375	42.0984	176.7465	6059.7340	6085.2303	6371.0455	43.0130	27,500
Krohling &Coelho [21]	0.8125	0.4375	42.0913	176.7465	6061.0777	6147.1332	6363.8041	86.4500	240,000
Coello &Montes [6]	0.8125	0.4375	42.0973	176.65405	6059.9463	6177.2533	6469.3220	130.9297	80,000
Gandomi [15]	0.8125	0.4375	42.09844	176.63659	6059.71433	6495.3470	6447.7360	502.693	N.A
Salimi [35]	0.8125	0.4375	42.098445	176.63659	6059.71433	6059.71433	6059.71433	6059.71433	50,000
iDE*a*Sm	0.778198	0.384665	40.321054	199.98023	5988.027532	5890.400037	5887.10836	1.82062166	20,000

* N.A: Not Available

6. Conclusion

In recent years, evolutionary algorithms attracted the attention of many researchers to develop efficient and new ways to solve a wide spectrum of benchmark problems. Those benchmark problems can be found in almost every science field as optimization problems. Finding this solution with a specified budget of function evaluations states a challenge for most algorithms. A new improved differential evolution algorithm with a novel adapted surrogate model is introduced in this paper. In this proposed surrogate model, the covariance matrix, which is utilized to find the coefficient correlations between the parameters, is used to adapt the *theta* parameter in the Kriging model at every generation. The proposed approach falls in the first class which uses an adaptive scheme to control the settings of *theta* parameter for Kriging model. This adaptation helps to mimic the original objective function, and hence assist the evolutionary search to find better solutions. Different chances using different control parameter settings are used to generate high quality offspring in each generation using the proposed surrogate model. The proposed model is embedded in L-SHADE algorithm. The goal is to advance the search to find better solutions using the surrogate model, and hence not counting the function evaluations. The proposed surrogate model is also used to enhance the quality of the best solution using a local search method that is based on interior-point method. The new algorithm, namely iDE*a*Sm, is validated using two benchmark sets: bound-constrained and real-world optimization problems. The simulation results show that iDE*a*Sm has a statistically good performance and a speedy convergence compared to other state-of-the-art algorithms. Also, the algorithm is tested using two well-known engineering design problems: welded beam and pressure vessel, and the results show the ability of the proposed algorithm to solve such kind of problems comparing to other best methods from the literature.

Appendix

Table A.1. A summary for functions description for bound-constrained benchmark set [40].

	Function		$f(X^*)$	Search Range
f_1	Sphere Function	$f_1(x)=\sum_{i=1}^{D} x_i^2$	0	$[-100,100]^D$
f_2	Generalized Rosenbrock Function	$f_2(x)=\sum_{i=1}^{D-1}(100(x_i^2-x_{i+1}^2)^2+(x_i-1)^2)$	0	$[-100,100]^D$
f_3	Ackley function	$f_3(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-$ $\exp\left(\frac{1}{D}\sqrt{\sum_{i=1}^{D}\cos(2\pi x_i)}\right)+20+e$	0	$[-32,32]^D$
f_4	Generalized Griewank Function	$f_4(x)=\sum_{i=1}^{D}\frac{x_i^2}{4000}-\prod_{i=1}^{D}\frac{x_i}{\sqrt{i}}+1$	0	$[-600,600]^D$
f_5	Generalized Rastrigin Function	$f_5(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$	0	$[-5.12,5.12]^D$
f_6	Non-continuous Rastrigin Function	$f_6(x)=\sum_{i=1}^{D}(y_i^2-10\cos(2\pi y_i)+10)$ $y_i=\begin{cases}x_i & \lvert x_i\rvert<0.5\\ \frac{\text{round}(2x_i)}{2} & \lvert x_i\rvert\geq 0.5\end{cases}$	0	$[-500,500]^D$
f_7	Schwefel's Function 2.26	$f_7(x)=418.9829\times D-\sum_{i=1}^{D}x_i\sin(\lvert x_i\rvert^{0.5})$	0	$[-500,500]^D$
f_8	Schwefel's Problem 1.2	$f_8(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2$	0	$[-100,100]^D$
f_9	Schwefel's Problem 1.2 with Noise in Fitness	$f_9(x)=\left(\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2\right)(1+0.4\lvert N(0,1)\rvert)$	0	$[-100,100]^D$
f_{10}	High Conditioned Elliptic Function	$f_{10}(x)=\sum_{i=1}^{D}(10^6)^{(i-1)/(D-1)}x_i^2$	0	$[-100,100]^D$
f_{11}	Weierstrass's Function	$f_{11}(x)=\sum_{i=1}^{D}\left(\sum_{k=0}^{k_{\max}}\left[a^k\cos(2\pi b^k(x_i+0.5))\right]\right)$ $-D\sum_{k=0}^{k_{\max}}\left[a^k\cos(2\pi b^k.0.5)\right]$	0	$[-0.5,0.5]^D$
f_{12}	Schwefel's Problem 2.22	$f_{12}(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	0	$[-10,10]^D$
f_{13}	Schwefel's Problem 2.21	$f_{13}(x)=\max_i\{\lvert x_i\rvert,1\leq i\leq D\}$	0	$[-100,100]^D$
f_{14}	Generalized Penalized Function 1	$f_{14}(x)=\frac{\pi}{D}\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2\left[1+10(\pi y_{i+1})\right]$ $+(y_D-1)^2+\sum_{i=1}^{D}u(x_i,10,100,4)$ $y_i=1+0.25(x_i+1)$ $u(x_i,10,100,4)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0 & ,-a\leq x_i\leq a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	0	$[-50,50]^D$

f_{15}	Generalized Penalized Function 2	$f_{15}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2\left[1+\sin^2(3\pi x_{i+1})\right] + (x_D - 1)\left[1+\sin^2(2\pi x_D)\right]\right\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	0	$[-50,50]^D$

References

[1] V.S. Aragon, S.C. Esquivel, C.A.C. Coello, A modified version of a T-Cell algorithm for constrained optimization problems, Int. J. Numer. Meth. Eng., vol. 84, pp. 351–378, 2010

[2] T. Back, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford, U.K.: Oxford Univ. Press, 1996

[3] D. Buche, N.N. Schraudolph, P. Koumoutsakos, "Accelerating evolutionary algorithms using fitness function models," in Proceedings of GECCO Workshop on Learning, Adaptation and Approximation in Evolutionary Computation, 2003, pp. 166–169

[4] W. Carpenter, J.F. Barthelemy, A comparison of polynomial approximation and artificial neural nets as response surface AIAA Technical Report, vol. pp. 92–2247,1992.

[5] C.A.C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. Comput. Methods Appl. Mech. Eng., vol. 191, no. 11, pp. 1245–1287, 2002

[6] C.A.C. Coello, E.M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Advan. Eng. Inform. vol. 16, pp. 193–203, 2002

[7] S. Das, S.S. Mullick, P.N. Suganthan, "Recent advances in differential evolution - An updated survey," Swarm and Evolutionary Computation, vol. 27, pp. 1-30, 2016

[8] S. Das, P. N. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems," Dept. Electron. Telecommun. Eng., Jadavpur Univ., Kolkata, India, Nanyang Technological Univ., Singapore, Tech. Rep., Dec. 2010

[9] K. Deb, Optimal design of awelded beam via genetic algorithms, Am. Inst. Aeronaut. Astronaut., vol. 11, pp. 2013–2015, 1991

[10] J. Derrac, S. Garcia, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation, vol. 1, pp. 3–18, 2011

[11] A. Diaz-Manriquez, G. Toscano-Pulido, W. Gomez-Flores, on the selection of surrogate models in the evolutionary optimization algorithms, in Proc. IEEE Congr. Evol. Comput., 2011, pp. 2155–2162

[12] S.M. Elsayed, T. Ray, R.A. Sarker, "Surrogate-assisted Differential Evolution Algorithm with Dynamic Parameters Selection for Solving Expensive Optimization Problems," in Proc. IEEE Congr. Evol. Comput., 2014

[13] M.S. Elsayed, A.R. Sarker, D.L. Essam, GA with a New Multi-Parent Crossover for Solving IEEE-CEC2011 Competition Problems, in Proc. IEEE Congr. Evol. Comput., 2011, pp. 1034 – 1040

[14] W. Gong, A. Zhou, Z. Cai, "A Multioperator Search Strategy Based on Cheap Surrogate Models for Evolutionary Optimization," IEEE Trans. Evol. Comput., vol. 19, no. 5, pp. 746–758, 2015

[15] A. H. Gandomi, X. S. Yang and A. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Engineering with Computers, vol. 29, pp. 17–35, 2003

[16] U. Halder, S. Das, D. Maity, A. Abraham, P. Dasgupta, Self Adaptive Cluster Based and Weed Inspired Differential Evolution Algorithm For Real World optimization, in Proc. IEEE Congr. Evol. Comput., 2011

[17] F.-z. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, Appl. Math. Comput., vol. 186, pp. 340–356, 2007

[18] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft Comput., vol. 9, pp. 3–12, 2003

[19] A. Kaveh, S. Talatahari, Hybrid charged system search and particle swarm optimization for engineering design problems, Int. J. Comput. Aided Eng. Softw. vol. 28, pp. 423–440, 2011

[20] J. Knowles, H. Nakayama, "Meta-modeling in multiobjective optimization," in *Multiobjective Optimization* (LNCS 5252). Berlin, Germany: Springer, 2008, pp. 245–284

[21] R.A. Krohling, L. dos Santos Coelho, Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems, IEEE Trans. Syst. Man Cybern. Part B: Cybern., vol. 36, pp. 1407–1416, 2006

[22] H. Kumar Singh, T. Ray, "Performance of a Hybrid EA-DE-Memetic Algorithm on CEC 2011 Real World Optimization Problems", in Proc. Congress on Evolutionary Computation, pp. 1322 - 1326, New Orleans, June 2011

[23] A. LaTorre, S. Muelas, J.-M. Pena, Benchmarking a Hybrid DE-RHC Algorithm on Real World Problems, in Proc. IEEE Congr. Evol. Comput., 2011

[24] Y.-X. Lei, J. Gou, C. Wang, W. Luo, Y.-Q. Cai, "Improved Differential Evolution With a Modified Orthogonal Learning Strategy," IEEE Access, vol 5, pp. 9699-9716, 2017

[25] D. Lim, Y. Jin, Y.-S. Ong, B. Sendhoff, "Generalizing surrogate assisted evolutionary computation," IEEE Trans. Evol. Comput., vol. 14, no. 3, pp. 329–355, Jun. 2010

[26] 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

[27] R. Mallipeddi, M. Lee, "An evolving surrogate model-based differential evolution algorithm," Appl. Soft Comput., vol. 34, pp. 770–787, 2015

[28] R. Mallipeddi, P.N. Suganthan, Ensemble Differential Evolution Algorithm for CEC2011 Problems, in Proc. IEEE Congr. Evol. Comput., 2011

[29] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," Appl. Soft Comput., vol. 11, no. 2, pp. 1679–1696, 2011

[30] X.Y. Nesterov, A. Nemirovskii, Y. Ye, Interior-point polynomial algorithms in convex programming. SIAM, vol. 13, 1994

[31] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Com-put., vol. 13, pp. 398–417, 2009

[32] X. Qiu, K.C. Tan, J.-X. Xu, "Multiple Exponential Recombination for Differential Evolution," IEEE Trans. Cybernetics, vol. 47, No. 4, 2017

[33] A. Ratle, "Kriging as a surrogate finess landscape in evolutionary optimization," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 15, pp. 37–49, 2001

[34] A. Ratle, "Accelerating the convergence of evolutionary algorithms by fitness landscape approximation," in Parallel Problem Solving from Nature - PPSN V, ser. Lecture Notes in Computer Science, vol. 1498, pp. 87–96, 1998

[35] H. Salimi, Stochastic Fractal Search: A powerful metaheuristic algorithm, Knowledge-Based Systems, vol. 75, pp. 1–18, 2015

[36] E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, J. Mech. Des., vol. 112, 1990

[37] T. Simpson, T. Mauery, J. Korte, F. Mistree, Comparison of response surfaceand Kriging models for multidisciplinary design optimization AIAA TechnicalReport, pp. 98–4755, 1998

[38] H.K. Singh, A. Isaacs, T. Ray, "A Hybrid Surrogate Based Algorithm (HSBA) to Solve Computationally Expensive Optimization Problems," in Proc. IEEE Congr. Evol. Comput., 2014

[39] R. Storn, K. Price, Differential evolution – a simple and efficient heuristicfor global optimization over continuous spaces, J. Glob. Optim., vol., 11, pp. 341–359, 1997

[40] 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 Sessionon Real-Parameter Optimization, Nanyang Technological University/KanGAL,IIT Kanpur, Singapore/India, May 2005

[41] R. Tanabe, A. Fukunaga, "Success-History Based Parameter Adaptation for Differential Evolution in Proc. IEEE Congr. Evol. Comput., 2013, pp. 71–78.

[42] R. Tanabe, A. Fukunaga, "Improving the Search Performance of SHADE Using Linear Population Size Reduction," in Proc. IEEE Congr. Evol. Comput., 2014, pp. 1658 - 1665

[43] G. Tomassetti, A cost-effective algorithm for the solution of engineering problems with particle swarm optimization, Eng. Optimiz. vol. 42, pp. 471–495, 2010

[44] P.M. Vasant, Meta-Heuristics Optimization Algorithms in Engineering, Business, Economics, and Finance, IGI Global, 2012, DOI: 10.4018/978-1-4666-2086-5

[45] P. Vasant, G.-W. Weber, V.N. Dieu, Handbook of Research on Modern Optimization Algorithms and Applications in Engineering and Economics, IGI Global, 2016, DOI: 10.4018/978-1-4666-9644-0

[46] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput., vol. 15, pp. 55-66, 2011.

[47] M. Yang, C. Li, Z. Cai, J. Guan, "Differential Evolution with Auto-Enhanced Population Diversity," IEEE Trans. Cybernetics, vol. 45, pp. 302 – 315, 2015

[48] J. Zhang, A.C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," IEEE Trans Evol Comput, vol. 13, no. 5, pp. 945–958, 2009

[49] S.X. Zhang, S.Y. Zheng, L.M. Zheng, "An Efficient Multiple Variants Coordination Framework for Differential Evolution," IEEE Trans. Cybernetics, vol. 47, no. 9, pp. 2780 – 2793, 2017

[50] Y.-Z. Zhou, W.-C. Yi, L. Gao, X.-Y. Li, "Adaptive Differential Evolution With Sorting Crossover Rate for Continuous Optimization Problems," IEEE Trans. Cybernetics, vol. 47, No. 9, pp. 2724 – 2735, 2017