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

A Surrogate-assisted Differential Evolution Algorithm with Dynamic Parameters Selection for Solving Expensive Optimization Problems

Saber M. Elsayed, T. Ray and Ruhul A. Sarker

Abstract— **In this paper, a surrogate-assisted differential evolution (DE) algorithm is proposed to solve the computationally expensive optimization problems. In it, the Kriging model is used to approximate the objective function, while DE employs a mechanism to dynamically select the best performing combinations of parameters (amplification factor, crossover rate and population size). The performance of the algorithm is tested on the WCCI2014 competition on expensive single objective optimization problems. The experimental results demonstrate that the proposed algorithm has the ability to obtain good solutions.**

Index Terms— **differential evolution, parameter selection, surrogate models, Kriging model**

I. INTRODUCTION

IN many real-world applications, there exist many computationally expensive mathematical or physical models. To solve these computationally expensive problems, an enormous number of fitness function evaluations (performance evaluations) are required during the evolution process when evolutionary algorithms (EAs) are used. Any time limitation in solving these problems would limit the search space exploration. To tackle this problem, recently, EAs based surrogate model have attracted much attention [1]. In such algorithms, a surrogate model is used to estimate the objective function by constructing an approximate model [2]. Over the last few decades, many surrogate models have been proposed, such as, Kriging, polynomials, and radial-basis-function networks (RBFN) [3].

Beside using surrogate model with EAs, it is well-known that the choice of control parameters in any EAs plays a critical role in its success [4]. In this paper, DE is used, as it has shown significant success in solving different numerical optimization problems (both constrained and unconstrained, and black-box) [5, 6] .In DE, one tedious technique, to find the best parameters combination, is a trial-and-error approach [7]. However, the best set of parameters is problem dependent, because of the variability of the underlying mathematical properties of the optimization problems. This means that a fixed set of control parameters that suits well for one problem or a class of problems does not assure that it will work well for another class, or a range of problems. Not only this, it is confirmed that a set of parameters that works well at the early stages of the evolution process may not perform well at the later stages and vice versa [4].

The idea of parameter adaptation was introduced at least two decades ago in the context of genetic algorithms [8]. In DE, many different mechanisms have been introduced to select and/or manage the dynamic changes of the control parameters. Based on how the control parameters are adapted, the mechanisms can be classified into three classes[9]: (1) *Deterministic Parameter Control* [10]; (2) *Adaptive Parameter Control* [11, 12]; and *Self-adaptive Parameter Control* [11-13]. Some of these algorithms have been applied to unconstrained problems, where it dynamically adapted either one of the three control parameters (crossover rate, amplification factor, or the population size), or two of them together (crossover rate and amplification factor). To the best of our knowledge, only a few algorithms reported in literature adapted all three control parameters together [14, 15]. In addition, existing investigations usually suggested a single set of parameters for all the problems under consideration. Note that some of the investigations, that determine the parameters using the traditional parametric analysis concept, require a huge number of trials.

Therefore, in this research, a surrogated assisted DE algorithm with dynamic selection is proposed for solving computationally expensive unconstrained problems. We introduce the algorithm as a surrogated-assisted DE with dynamic parameters selection (Sa-DE-DPS). In it, an initial sample is generated. A Kriging model is then used to build a model based on the initial sample. Once the model is built, DE-DPS is run to maximize the expected improvement. The best solution obtained from DE-DPS is evaluated on the exact fitness function and then added to the initial sample and a new model is rebuilt and so on. For a further improvement, after a predefined number of fitness evaluations, a local search is conducted. For clarity, in DE-DPS, three sets of parameters are considered: the first set is for the amplification factor, the second is for the crossover rate, while the third is for the population size. Each individual in the population is assigned a random combination of amplification factor (*F)* and crossover rate (*Cr)*. The success rate of each combination is recorded for a certain number of generations and the better performing combinations are applied for a number of subsequent generations. This process is recognized as a cycle. Based on the success rate, the number of combination is reduced in subsequent cycles. At the beginning of each cycle, the success rates of the current combinations are re-initialized to zero and after every few cycles, the process restarts with all combination of parameters.

The performance of the proposed algorithm is tested the CEC2014 competition on computationally expensive single objective numerical optimization [16], with different mathematical properties. From the results obtained, the proposed algorithm shows consistently ability to obtain good results.

The authors are with the School of Engineering and Information Technology, University of New South Wales, ADFA Campus, Canberra 2600, Australia (e-mails: {s.elsayed, t.ray and r.sarker}@adfa.edu.au)

This paper is organized as follows. After the introduction, section II presents the DE algorithm with an overview of its parameters. Section III describes the surrogate model used in this research, while Section IV describes the design of the proposed algorithm. The experimental results and the analysis of those results are presented in section V. Finally, conclusions are given in section VI.

II. DIFFERENTIAL EVOLUTION

In this section, the commonly used DE operators and parameters are discussed.

Te stat with, we define the key terms that are used in this section. A target vector ($\vec{x}_{z,t}$) is a parent vector in generation *t* of an individual *z*. A mutant vector ($\vec{v}_{z,t}$) is the vector obtained through the mutation operation, which is also known as the *donor* vector. A trial vector ($\vec{u}$) is an offspring which is obtained by recombining the mutant vector with the parent vector.

A. Mutation

In the simplest form of mutation, $\vec{v}_{z,t}$ is generated by multiplying the amplification factor F by the difference of two random vectors selected from the current population, and the result is added to another third random vector from the current population.

$$\vec{v}_{z,t} = \vec{x}_{r_1,t} + F\left(\vec{x}_{r_2,t} - \vec{x}_{r_3,t}\right) \quad (1)$$

where r_1, r_2, r_3 are random integer numbers [1,*N*], $r_1 \neq r_2 \neq r_3 \neq z$, x is a decision vector, *PS* is the population size and *t* is the current generation. Using a technique to handle bound constraints is indeed essential.

This operation enables DE to explore the search space and maintain diversity. There are many strategies for mutation, such as DE/rand-to-best/2 [12], rand/2/dir [17], DE/current-to- best/1[18], and DE/Current-to-pbest [19]. For more details, readers are referred to Das and Suganthan [5].

B. Crossover

In DE, two crossover operators (exponential and binomial) are commonly used. These crossover operators are briefly discussed below.

In an exponential crossover, an integer *l* is randomly chosen within the range [20], where D is the number of decision variables. This integer acts as a starting point in $\vec{x}_{z,t}$, from where the crossover or exchange of components with $\vec{v}_{z,t}$ starts. Another integer *L* is chosen from the interval [1, *D-l*] [5].

The trial vector ($\vec{u}$) is formed by inheriting the values of variables in locations *l* to $l + L$ from the mutant vector and the remaining ones from the parent vector.

The binomial crossover is performed on each of the j^{th}variables whenever a randomly picked number (between 0 and 1) is less than or equal to a crossover rate (*Cr*). The generation number is indicated here by *t*. In this case, the number of parameters inherited from the donor has a (nearly) binomial distribution.

$$u_{zj,t} = \begin{cases} v_{zj,t}, & if\ (rand \leq Cr\ or\ j = j_{rand}) \\ x_{zj,t}, & otherwise \end{cases} \quad (2)$$

where $rand$ is a uniform random number $\in [0,1]$, and $j_{rand} \in \{1, 2, \ldots, D\}$ is a randomly chosen index, which ensures $\vec{u}_{z,t}$ gets at least one component from $\vec{v}_{z,t}$

During the last two decades, adapting DE parameters has taken a great attention. For example, Abbass [20] proposed a self-adaptive operator (crossover and mutation) for multi-objective optimization problems, where the amplification factor F is generated using a Gaussian distribution $N(0,1)$. This technique has been modified in [21]. Zaharie [22] proposed a parameter adaptation strategy for DE (ADE) based on the idea of controlling the population diversity, and implemented a multiple population approach.

Qin *et al.* [12] proposed a novel differential evolution algorithm (SaDE), where the choice of the learning strategy and the two control parameters *F* and Cr are not required to be pre-specified. The parameter *F*, in SaDE, is approximated by a normal distribution *N* (0.5, 0.3), and truncated to the interval (0, 2]. Such an approach could maintain both the intensification (with small *F* values) and diversity (with large *F* values) during the course of search. The crossover probabilities were randomly generated according to an independent normal distribution with mean Cr_m and standard deviation 0.1. The Cr_m values remain fixed for five generations before the next re-generation. Cr_m was initialized to 0.5, and it was updated every 25 generations based on the recorded successful *Cr* values since the last Cr_m update.

Using fuzzy logic controllers, Liu and Lampinen [23] presented a fuzzy adaptive differential evolution, whose inputs incorporated the relative function values and individuals of successive generations to adapt the parameters for mutation and crossover. Brest *et al.* [11] proposed a self-adaptation scheme for the DE control parameters, known as jDE. The control parameters were adjusted by means of evolution of *F* and *Cr.* In jDE, a set of *F* and *Cr* values was assigned to each individual in the population, augmenting the dimensions of each vector.

III. SURROGATE MODELS

Surrogate models are (statistical) models that are built to approximate the actual model, i.e. if the true model of a problem can be presented as ($y = f(x)$), then a surrogate model is an approximation of the form ($\hat{y} = \hat{f}(x)$), such that $y = \hat{y} + \varepsilon$, where ε is a difference value between the two functions. Hence, using such model, instead of high fidelity optimization simulations, can reduce the computational cost [24].

There exist a range of procedures for building surrogate models, such as: Kriging models [25], radial basis function networks [26], and support vector machines [27]. In this paper, we consider the Kriging model as in it a confidence interval of the estimation can be obtained without much additional computational cost [24].

Kriging exploits the spatial correlation of data in order to predict the shape of the objective function [28]. The correlation function relies on the linear regression model

and the Gaussian correlation model, as shown in (3) and (4), respectively.

$$\hat{y}(x) = \sum_{k=1}^{m} \beta_k f_k(x) + \varepsilon(x) \quad (3)$$

$$R\left(\varepsilon(x^i), \varepsilon(x^j)\right) = \Pi_{k=1}^{N} e^{-\theta_k \left|x_k^i - x_k^j\right|^{p_k}} \quad (4)$$

where $\beta_k f_k(x)$ is the global function, and $\varepsilon(x)$ Gaussian noise, θ_k the correlation amongst the data in k-direction and $p_k = 2$ corresponding to smooth functions and values near 1 corresponding to less smoothness [29]. The maximum likelihood estimation optimizes the value of θ and then the correlation model is brought into the regression model to evaluate the function with the best linear unbiased predictor [30].

The expected improvement utility function [29] is used to select multiple designs. This utility function is based on the mean square error (MSE) generated by the Kriging model. Therefore, expected improvement can be seen as:

$$EI = \left(f_{best} - \hat{y}(x)\right)\Phi\left(\frac{f_{best} - \hat{y}(x)}{s(x)}\right) + s(x)\phi\left(\frac{f_{best} - \hat{y}(x)}{s(x)}\right) \quad (5)$$

where f_{best} is the best objective value obtained, $s(x)$ is the root mean squared error in the predicted objective function $\hat{y}(x)$, $\Phi(\cdot)$ and $\phi(\cdot)$ are the standard normal density and distribution function, respectively.

IV. Surrogated-assisted DE with dynamic parameters selection

In this section, the proposed algorithm (Sa-DE-DPS) is described.

A. The Algorithm

The general framework of the proposed methodology is described in Algorithm I.

To start with, an initial sample of an even size (N) is generated by a symmetric Latin hypercube design. Using this sample of solutions, the Kriging model is then built. Note that the parameters of the Kriging model are generated by a simple DE (DE/φ rand/1/bin [31]), such that:

$$u_{zj,t} = \begin{cases} x_{\varphi j,t} + F_z.\left(x_{r_1 j,t} - x_{r_2 j,t}\right), if\ (rand \le Cr_z\ or\ j = jrand) \\ x_{zj,t}, \qquad otherwise \end{cases} \quad (6)$$

where Cr_z is randomly selected from {0.4, 0.9 and 0.99}, while $F_z \in [0.4, 0.95]$, φ is an integer random number $\in \left[1, \frac{N}{2}\right]$, and r_1 and r_2 are random integer numbers $\in [1, N]$, $r_1 \neq r_2 \neq z$. Based on the model built, DE-DPS is evolved to find the maximum expected improvement, and the only best solution found is added to the initial sample (after evaluating it on the true fitness function, i.e. increase the objective function evaluations by one) and re-built the model again. It must be mentioned here that if $N > \upsilon$, then the best υ individuals in the sample are used to built the model. In addition, to exploitation searching, a local search is performed, up to sqpFEs, on the best solution found so far. It is worthy to mention here that the local search is applied only once, as a limited number of fitness function evaluations (FFEs) is used, in the competition, as a stopping criterion. The process continues till a termination criterion is met. Note that the local search is applied only once during the entire algorithm process.

ALGORITHM I: General framework of Sa-DE-DPS

Step 1: Generate a population of solutions using the symmetric Latin hypercube design. Calculate the fitness values of all points, based on the true fitness function and hence update the fitness function evaluations.
Step 2: Create the Kriging model. The parameters of the Kriging model are identified using DE.
Step 3: Use DE-DPS to find the best x location that gives the maximum expected improvement and evaluate the expensive function there only, and hence increase FFEs by **1**.
Step 4: If condition is met, apply local search on the best solution found so far and update the number of FFE.
Step 5: Add this solution to the list and retrain the Kriging model and go to **Step 2**.

B. DE-DPS

Here DE-DPS used to find the maximum expected improvement is presented.

The motivation behind DE-DPS is to find the most appropriate parameters (*F*, *Cr*, and *PS*) during the evolution process. The sets for the parameters *F*, *Cr*, and *PS* are defined as F_{set}, Cr_{set} and PS_{set}, where, $F_{set} = \{F_1, F_2, \dots, F_{nf}\}$, $Cr_{set} = \{Cr_1, Cr_2, \dots, Cr_{ncr}\}$, $PS_{set} = \{PS_1, PS_2, \dots, PS_{nps}\}$. Here PS_i is assumed to be larger than PS_{i-1}, $\forall\ i = nps, nps-1, \dots, 2$, and nf, ncr and nps refer to the cardinality of the set of amplification factors, crossover rates, and population sizes, respectively. Note that the population size (PS_{i-1}) is not only smaller than (PS_i), but also a subset of PS_i. Similarly, PS_{i-2} is a subset of PS_{i-1} and so on.

The pseudo code of the algorithm is presented in Algorithm II. In the first step, PS_{nps} (i.e. the population with the largest size considered in this paper) random individuals are generated within the variable bounds. Each individual in the population ($\vec{x}_z$) is assigned a random *F* (F_z) and a random *Cr* (Cr_z). The number of combinations ($tot.com$) for *F* and *Cr* is equal to $nf \times ncr$. Note that there are nps population sizes.

For each z^{th} individual in the population, a new offspring is generated first via a mutation operator which is further modified via a crossover operation. To perform mutation, three individuals are used, two of which are randomly selected from the population, while the third base parent is selected from between [a, b], where a is set to 1 and b is set to $\frac{PS}{4}$, in this study. Following the mutation operation, the crossover is performed between the individual generated via the above mutation process and the z^{th} individual in the population. Mathematically, the process can be presented as follows:

$$u_{zj,t} = \begin{cases} x_{zj,t} + F_z\left(x_{\varphi j,t} - x_{zj,t}\right) + F_z\left(x_{r_1 j,t} - x_{r_2 j,t}\right) \\ \qquad\qquad , if\ (rand \le Cr_z\ or\ j = jrand) \\ x_{zj,t}, \qquad\qquad otherwise \end{cases} \quad (7)$$

here, φ is a random integer number within a range [a, b].

ALGORITHM II. DE-DPS ALGORITHM

STEP 1: At generation $t = 1$, generate an initial random population of size PS_{nps}.

STEP 2: Set $PS_{set} = \{PS_1, PS_2, \dots, PS_{nps}\}$, set $period = 0$, *PS_period=0,* $i = nps$ and $PS = PS_i$

STEP 3: Set $F_{set} = \{F_1, F_2, \dots, F_{nf}\}$, $Cr_{set} = \{Cr_1, Cr_2, \dots, Cr_{ncr}\}$

STEP 4: Generate new offspring as follows:

- **4.1** Each individual is assigned a random combination (rc) of parameters, $rc \in rc_{set}$, and rc_{set} is the combination of all F_{set} and Cr_{set}.
- **4.2** Generate the offspring vector $\vec{v}_z$, using mutation and binomial crossover operators as in (7), and update the FFEs.
- **4.3** If $\vec{v}_z$ is better than its parent, then: $com.suc_{rc} = com.suc_{rc} + 1$
- **4.4** $period = period + 1$;
- **4.5** $PS\_period = PS\_period + 1$;

STEP 5: If *period*% CS=0 and *period*<(*η*×*CS*):

- **5.1** Select the best half combination to be used in the evolution process [based on the rankings using equation (8)] and update rc_{set}.
- **5.2** Set each $com.suc_{rc} = 0$, and go to **Step 4.**

Else If $period\%(\eta \times CS) = 0$

- **5.3** Set each $com.suc_{rc} = 0$ and $period = 0$

STEP 6: If $PS\_period \% CS = 0$ **and** $i > 0$

- **6.1** calculate $Rank_{PS_i}$ using:

$$Rank_{PS_i} = \frac{\sum_1^{CS} \sum_{cy=1}^{tot.com} com.suc_{cy}}{PS_i}$$

If $i < nps$

- **6.2** archive the worst $(PS_i - PS_{i-1})$ individuals
- **6.3** **Set** $i = i - 1$
- **6.4** Set $PS = PS_i$

STEP 7: If *i =0 and PS_period= nps* × CS

- 7.1 Set *PS* to the one with the best $Rank_{PS_i}$

STEP 8: If $PS\_period = \eta \times CS$

- 8.1 Set *PS_period=0,* $i = nps$ and $PS = PS_i$
- 7.2 Use individuals from the archive as required.
- 7.3 Clear the archive

STEP 8: Stop if the termination criterion is met; **else,** set $t = t + 1$ **and** go to **STEP 4**.

The introduction of these parameters (φ, *a,* and *b)*, and if $\varphi = 1$, the abovementioned mutation will be DE/current-best/1 mutation, $z \in \{1, 2, \dots, PS\}$ and r_1 and r_2 are random integer numbers $\in [1, PS]$, $r_1 \neq r_2 \neq z$.

If the new offspring is better than its parent i.e. the i^{th} individual in the population, it will be accepted and the success of a combination rc $(com.suc_{rc})$ is increased by one, i.e. $com.suc_{rc} = com.suc_{rc} + 1$, where $rc = 1, 2, \dots, tot.com$.

The above process is repeated for *CS* generations. At the end of *CS* generations, the better performing PS_{nps-1} individuals are kept in the population, while the remaining individuals are transferred to an archive. The number of combinations of F and Cr values is also reduced to half i.e. the better combinations of F and Cr are preserved based on the success of the combination. The ranking of any combination is calculated using the following equation.

$$Rank_{rc} = \frac{com.suc_{rc}}{the\ number\ of\ individuals\ used\ a\ combination\ rc} \quad (8)$$

where a higher value of $Rank_{rc}$ is a better performing combination.

Individuals in the population of size PS_{nps-1} are randomly assigned F and Cr values from this reduced list. The process of evolution uses the same mutation and crossover strategy and is allowed to evolve for *CS* generations. This process is repeated until all population sizes are considered. At the end of this stage, the performance ranking of all population sizes are computed, using the equation shown in Step 6.1 in Table I, to decide the appropriate population size. The performance ranking of any population size represents the average performance per individual in the population for all parameter combinations over *CS* generations.

In the event PS_{nps} is selected, the best individuals from the archive are added to make the current population size (PS_i) equal to PS_{nps}. The number of combinations of F and Cr values are also reduced to half based on the success as described earlier. The selected population of size (PS_i) is allowed to evolve for $(\eta - nps) \times CS$ generations wherein after every CS generations, the number of combinations of F and Cr are reduced to half until the number of combinations reaches 1. The above steps are referred to a cycle. At the end of each cycle, the success tables and the archives are reset to null, the total number of combinations is reset to *tot.com*, and the population size is reset to PS_{nps}. The cycles continue until the termination criterion is met. To clarify, an example with two population sizes (100 and 75, *i.e. nps* = 2) and 64 combinations of rc is considered. The population size 100 will evolve for *CS* generations with 64 combinations, then the population size 75 will evolve for *CS* generations with 32 combinations, and finally the selected population size (either 100 or 75) will be fixed for the next $(\eta - 2) \times CS = 4CS$ generations.

C. Discussions on Related Issues

In this section, we discuss few issues relevant to the algorithm design and implementation.

In the evolution process, for a given problem, the relative performance of each combination may vary with the progression of generations. This behavior means that one combination may work well at the early (or some) stages of the search process and may poorly perform at the later (or some other) stages, or vice-versa. So, it is inappropriate to give equal emphasis on all of the combinations throughout the entire process of evolution. To give a higher emphasis on the better performing combinations in a given stage of the evolution process, it is proposed that the random assignment of the parameter combinations is applied for a fixed number of generations (say *CS*).

The parameter combinations are assigned randomly to individuals without replacement. That means, one combination will be assigned strictly to one individual if the population size is less than or equal to the number of combinations. If the population size is larger than the number of combinations, all combinations are assigned to at least one individual. Depending on the number of combinations and population size, one combination may be assigned to more than one individual and there is a possibility that some combinations may not be assigned at

all. The ranking of any assigned combination is calculated using (8) and the ranking of any unassigned combination is set to zero.

D. Sequential Quadratic Programming (*SQP*)

SQP has become a powerful method for solving for constrained optimization problems (COPs) [32], a COP can be represented as:

$$minmize\ f(x)$$

subject to:

$$\begin{aligned} b(x) &\geq 0, \\ c(x) &= 0 \end{aligned} \tag{9}$$

However, it can be successfully applied for unconstrained problems. The main idea of SQP is to model a problem at the current point x_k by a quadratic sub-problem of (9), such as:

$$min\ f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T H_k d$$

subject to

$$\begin{aligned} b(x_k) + \nabla b(x_k)^T d &\geq 0 \\ c(x_k) + \nabla c(x_k)^T d &= 0 \end{aligned} \tag{10}$$

and to use the solution of this sub-problem to find the new point x_{k+1}.

As SQP may be viewed as an extension of Newton and quasi-Newton methods to the constrained optimization setting, SQP methods could share the characteristics of Newton-like methods, such as when the iterates are close to the solution a rapid convergence can be achieved, when the iterates are far from a solution a possible eccentric behaviour can be happened that needs to be carefully controlled [33]. Note that this research is only for unconstrained problems.

V. EXPERIMENTAL RESULTS

In this section, the performance of the proposed algorithm is discussed and analyzed by solving a set of problems presented in the CEC2014 competition on computationally expensive single objective numerical optimization [16], which contains 8 test problems with 10, 20 and 30 dimensions, with different mathematical properties (unimodal, multi-modal, continuous, discrete, separable and non-separable). To add to this, the optimal solutions ($f^* = 0$) are shifted and/or rotated. The algorithm was run 20 times for each test problem, where the stopping criterion was to run for up to 500, 1000 and 1500 FFEs, respectively, or $f(x_{best}) - f(x^*)) \leq 10^{-8}$.

The algorithm was coded using Matlab R2012b, and was run on a PC with a 3.4 GHz Core I7 processor with 16 GB RAM, and windows 7. The parameter values are shown in Table I. The detailed results (best, median, worst, mean and standard deviations) of $(f(x_{best}) - f(x^*))$ are shown in Table II.

From results obtained, it is clear that Sa-DE-DPS's was robust in F01-F03. Sa-DE-DPS's performance was good for F04, while it was very close to the optimal solutions for F05 and F06. Sa-DE-DPS's performance in F07-F09 did not differ much from those results of F04-F07, as Sa-DE-DPS was able to reach the optimal solutions in 10D and obtained close results in the 20D and 30D instances, i.e. F08 and F09, respectively.

TABLE I. DETAILS OF ALL PARAMETERS VALUES

Kriging: $N = 25$, points used to build the model are the best $min\ (N, \upsilon)$ solutions, where $\upsilon = 50$, the initial boundary for θ is $[-3,3]$, initial population for DE used in likelihood estimation to optimize θ is 50 and runs for 50 generations, while the DE parameters are shown in IV.A.

DE-DPS: $F_{set} = \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99\}$, $Cr_{set} = \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99\}$), $PS_{set} = \{75,100\}$, CS=25, and $\eta = 4$, stopping criteria are to run it up to 10,000D fitness evaluations or the best solution does not change for 100 generations, to find the maximum expected improvement.

SQP: sqpFEs = 25D, and done only once after 10D fitness evaluations.

Based on the results reached in 4th problem, Sa-DE-DPS was able to obtain the optimal solution for 10D (F10), but not over all runs, while its average performance in 20D (F11) was good, and the results were far from the optimal solution in the 30D (F12).

For the 5th problem, although the results obtained were not too far from the optimal solution, Sa-DE-DPS was found to converge to a single local solution over all runs.

Sa-DE-DPS's in the 6th problem was quite good, as it converged to a very close solution in 10D (F16) in all runs, and was consistently able to reach the optimal solution for 20D (F17) and 30D (F18).

In regards to 7th problem, Sa-DE-DPS was able to obtain a very close solution to the optimal in the 10D instances (F19), while all results were not too far from the optimal in 20D (F20) and 30D (F21). Similarly, in the last problem, Sa-DE-DPS obtained not too far solutions from the optimal results for all dimensions.

The convergence plots of the proposed algorithm for all test problems with 10 and 20D are presented in Fig.1. It appears from this figure that using SQP on the true function may lead the algorithm to get trapped in local optima, but this not the case for the unimodal problems as the valleys of the optimal solutions are rather steep.

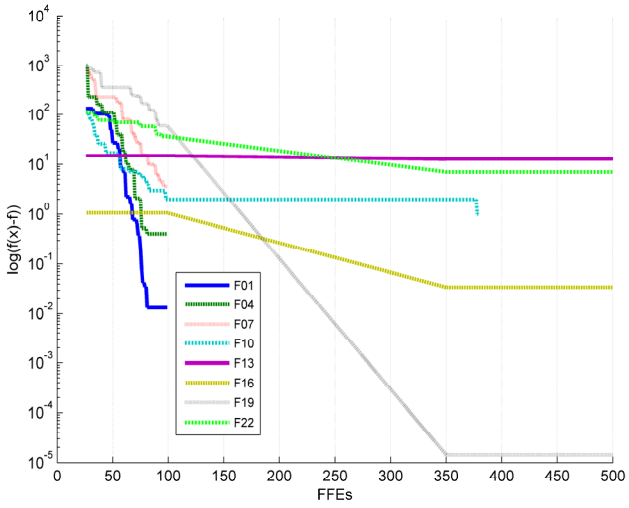
To this end, the complexity of the proposed algorithm is calculated based on all problem dimensions. A summary of the results is shown in Table III.

TABLE III. COMPUTATIONAL COMPLEXITY

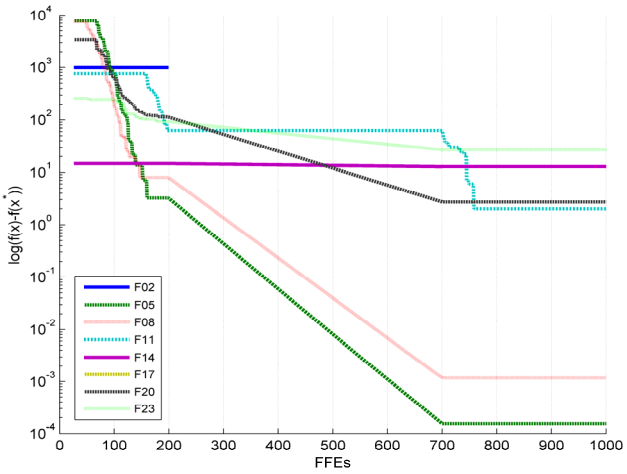
Func.	$\frac{\hat{T}_1}{T_0}$	Func.	$\frac{\hat{T}_1}{T_0}$
F01	6738.802	**F13**	2611.59
F02	12306.46	**F14**	5767.83
F03	17497.59	**F15**	8819.89
F04	4995.90	**F16**	11641.44
F05	52275.51	**F17**	17010.39
F06	53535.52	**F18**	43071.99
F07	3435.38	**F19**	16276.15
F08	14623.68	**F20**	53015.62
F09	55018.92	**F21**	114725.8
F10	5977.595	**F22**	15661.55
F11	26966.43	**F23**	51592.41
F12	55144.1	**F24**	117955.5

TABLE III. RESULTS FOR 10D

	Best	Median	Worst	Mean	Std
F01	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**
F02	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**
F03	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**
F04	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**
F05	1.349093E-03	1.298536E-02	2.224439E+00	3.006592E-01	6.471878E-01
F06	4.819746E+00	1.068824E+01	2.369976E+01	1.196645E+01	4.300321E+00
F07	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**
F08	1.183037E-03	1.960552E-02	6.860622E-01	5.699681E-02	1.494529E-01
F09	1.460981E+01	1.460981E+01	5.175838E+01	1.908386E+01	1.033298E+01
F10	**0.000000E+00**	1.000000E+00	5.000000E+00	9.000000E-01	1.209611E+00
F11	2.000000E+00	9.500000E+00	3.000000E+01	1.030000E+01	6.122435E+00
F12	4.900000E+01	9.865000E+02	1.005000E+03	8.625000E+02	2.934618E+02
F13	1.277693E+01	1.277693E+01	1.277693E+01	1.277693E+01	**0.000000E+00**
F14	1.275559E+01	1.275559E+01	1.275559E+01	1.275559E+01	**0.000000E+00**
F15	1.176758E+01	1.176758E+01	1.176758E+01	1.176758E+01	**0.000000E+00**
F16	3.450218E-02	3.450218E-02	3.450218E-02	3.450218E-02	**0.000000E+00**
F17	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**
F18	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**
F19	1.422114E-05	1.830841E+00	7.557295E+00	3.145679E+00	3.116599E+00
F20	2.705425E+00	1.569212E+01	6.893425E+01	2.092630E+01	2.052821E+01
F21	2.700339E+01	2.700339E+01	3.791911E+01	2.868610E+01	3.546546E+00
F22	6.964708E+00	1.840671E+01	3.979822E+01	1.974990E+01	8.104135E+00
F23	2.686387E+01	4.974788E+01	7.763040E+01	4.830880E+01	1.332292E+01
F24	7.896960E+01	1.030690E+02	1.224382E+02	1.025888E+02	7.163882E+00



(a) 10D



(b) 20D

Fig. 1. Convergence plots of Sa-DE-DPS obtained for each test problem with 10 and 20D. The y-axis is in a log scale of $f(x_{best}) - f(x^*)$) and the difference is set to 0 if it is $\leq 10^{-8}$. FFEs refer to the maximum number of fitness evaluations and a solution

VI. CONCLUSIONS AND FUTURE WORK

During the last few decades, using evolutionary algorithms for solving optimization problems has shown good performance. However, in solving computationally expensive problems, EAs suffer from excessive evaluation of the objective function of a problem on hand. Therefore, using surrogate models to built approximate models of the objective functions has taken much attention during the least decades.

In this paper, a surrogate-assisted differential evolution was proposed, in which a Kriging model was employed to built an approximation model of the objective function during the evolution process, while DE was used to optimize the approximated model. In DE, three sets of parameter values were initialized one each for the amplification factor, crossover rate and population size. For a defined number of generations, each individual in the population was assigned to a random combination, and the normalized success for each combination was recorded. Subsequently, the number of combinations was reduced until a restart point, where the success counters were reset. To add to this, a local search was applied to exploit the search space on the true objective function

The performance of the proposed algorithm was tested on the WCCI2014 competition on computationally expensive single objective numerical optimization and showed good performance.

For future work, we would like to analyze each parameter of the proposed algorithm and test it on real-world applications.

REFERENCES

[1] Y. S. Ong, P. B. Nair, and A. J. Keane, "Evolutionary optimization of computationally expensive problems via surrogate modeling," *AIAA journal,* vol. 41, pp. 687-696, 2003.

[2] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation,* vol. 1, pp. 61-70, 2011.

[3] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing,* vol. 9, pp. 3-12, 2005.

[4] R. Sarker, S. Elsayed, and T. Ray, "Differential Evolution with Dynamic Parameters Selection for Optimization Problems," *Evolutionary Computation, IEEE Transactions on,* vol. PP, pp. 1-1, 2013.

[5] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art," *IEEE Transactions on Evolutionary Computation,* vol. 15, pp. 4-31, 2011.

[6] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking: Noiseless functions definitions," INRIA, Tech. Rep. 2009.

[7] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, "Self-Adaptive Multimethod Search for Global Optimization in Real-Parameter Spaces," *IEEE Transactions on Evolutionary Computation,* vol. 13, pp. 243-259, 2009.

[8] L. Davis, "Adapting operator probabilities in genetic algorithms," presented at the Proceedings of the third international conference on Genetic algorithms, George Mason University, United States, 1989.

[9] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*: Springer, 2003.

[10] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Ann Arbor. Michigan: University of Michigan Press, 1975.

[11] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems," *IEEE Transactions on Evolutionary Computation,* vol. 10, pp. 646-657, 2006.

[12] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization," *IEEE Transactions on Evolutionary Computation,* vol. 13, pp. 398-417, 2009.

[13] V. L. Huang, A. K. Qin, and P. N. Suganthan, "Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization," in proceeding *IEEE Congress on Evolutionary Computation*, 2006, pp. 17-24.

[14] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Computing,* vol. 10, pp. 673-686, 2006/06/01 2006.

[15] J. Brest and M. Sepesy Maučec, "Population size reduction for the differential evolution algorithm," *Applied Intelligence,* vol. 29, pp. 228-247, 2008/12/01 2008.

[16] Q. C. a. Q. Z. B. Liu, J. J. Liang, P. N. Suganthan, B. Y. Qu, "Problem Definitions and Evaluation Criteria for Computationally Expensive Single Objective Numerical Optimization," Computational Intelligence Laboratory and Nanyang Technological University, Zhengzhou and Singapore, Tech. Rep. 2013.

[17] V. Feoktistov and S. Janaqi, "Generalization of the strategies in differential evolution," in proceeding *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 2004, p. 165.

[18] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Berlin: Springer, 2005.

[19] Z. Jingqiao and A. C. Sanderson, "JADE: Adaptive Differential Evolution With Optional External Archive," *IEEE Transactions on Evolutionary Computation,* vol. 13, pp. 945-958, 2009.

[20] H. A. Abbass, "The self-adaptive Pareto differential evolution algorithm," in proceeding *IEEE Congress on Evolutionary Computation.*, 2002, pp. 831-836.

[21] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Multi-operator based evolutionary algorithms for solving constrained optimization Problems," *Computers and Operations Research,* vol. 38, pp. 1877-1896, 2011.

[22] D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms," in proceeding *the 9th International Conference on Soft Computing*, 2003, pp. 41–46.

[23] J. Liu and J. Lampinen, "A Fuzzy Adaptive Differential Evolution Algorithm," *Soft Computing - A Fusion of Foundations, Methodologies and Applications,* vol. 9, pp. 448-462, 2005.

[24] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset, "A rigorous framework for optimization of expensive functions by surrogates," *Structural optimization,* vol. 17, pp. 1-13, 1999/02/01 1999.

[25] D. Buche, N. N. Schraudolph, and P. Koumoutsakos, "Accelerating evolutionary algorithms with gaussian process fitness function models," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 35, pp. 183-194, 2005.

[26] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural computation,* vol. 3, pp. 246-257, 1991.

[27] J. A. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: robustness and sparse approximation," *Neurocomputing,* vol. 48, pp. 85-105, 2002.

[28] S. Xiao, M. Rotaru, and J. K. Sykulski, "Adaptive Weighted Expected Improvement With Rewards Approach in Kriging Assisted Electromagnetic Design," *Magnetics, IEEE Transactions on,* vol. 49, pp. 2057-2060, 2013.

[29] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization,* vol. 13, pp. 455-492, 1998.

[30] L. Lebensztajn, C. A. R. Marretto, M. C. Costa, and J.-L. Coulomb, "Kriging: a useful tool for electromagnetic device optimization," *Magnetics, IEEE Transactions on,* vol. 40, pp. 1196-1199, 2004.

[31] R. A. Sarker, S. M. Elsayed, and T. Ray, "Differential Evolution with Dynamic Parameters Selection for Optimization Problems," *IEEE Transactions on Evolutionary Computation, in press,* 2013.

[32] M. Powell, "A fast algorithm for nonlinearly constrained optimization calculations.," in *Numerical Analysis*. vol. 630, G. Watson, Ed., ed: Springer Berlin / Heidelberg, 1978, pp. 144-157.

[33] P. T. Boggs and J. W. Tolle, "Sequential Quadratic Programming," *Acta Numerica,* vol. 4, pp. 1-51, 1995.