Recent Advances in Differential Evolution – An Updated Survey

Swagatam Das[1], Sankha Subhra Mullick[1], and P. N. Suganthan[2]

[1]Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata-700 108, India.
[2]School of Electrical and Electronics Engineering, Nanyang technological University, Singapore.

E-mails: swagatam.das@isical.ac.in, mullicksankhasubhra@gmail.com, epnsugan@ntu.edu.sg

***Abstract*—Differential Evolution (DE) is arguably one of the most powerful and versatile evolutionary optimizers for the continuous parameter spaces in recent times. Almost 5 years have passed since the first comprehensive survey article was published on DE by Das and Suganthan in 2011. Several developments have been reported on various aspects of the algorithm in these 5 years and the research on and with DE have now reached an impressive state. Considering the huge progress of research with DE and its applications in diverse domains of science and technology, we find that it is a high time to provide a critical review of the latest literatures published and also to point out some important future avenues of research. The purpose of this paper is to summarize and organize the information on these current developments on DE. Beginning with a comprehensive foundation of the basic DE family of algorithms, we proceed through the recent proposals on parameter adaptation of DE, DE-based single-objective global optimizers, DE adopted for various optimization scenarios including constrained, large-scale, multi-objective, multi-modal and dynamic optimization, hybridization of DE with other optimizers, and also the multi-faceted literature on applications of DE. The paper also presents a dozen of interesting open problems and future research issues on DE.**

***Index Terms*— Differential evolution; continuous evolutionary optimization; numerical optimization; parameter adaptation; recombination.**

1. Introduction

In an attempt to find the global optimum of non-linear, non-convex, multi-modal and non-differentiable functions defined in the continuous parameter space ($\subseteq \mathbb{R}^d$), Storn and Price proposed the Differential Evolution (DE) (Storn and Price, 1995; 1997) algorithm in 1995. Since then, DE and its variants have emerged as one of the most competitive and versatile family of the evolutionary computing algorithms and have been successfully applied to solve numerous real world problems from diverse domains of science and technology (Das and Suganthan, 2011; Neri and Tirronen, 2010). Starting with a uniformly random set of candidate solutions sampled from the feasible search volume, every iteration (commonly known as *generation* in evolutionary computing terminology) of DE operates through the same computational steps as employed by a standard Evolutionary Algorithm (EA). However, DE differs markedly from the well-known EAs like Evolution Strategies (ESs) and Evolutionary Programming (EP) in consideration of the fact that it mutates the base vectors (secondary parents) with scaled difference(s) of the distinct members from the current population. As iterations pass, these differences tend to adapt to the natural scales of the objective landscape. For example, if the population becomes compact in one variable but remains widely dispersed in another, the difference vectors sampled from it will be smaller in the former variable, yet larger in the latter. This automatic adaptation significantly improves the search moves of the algorithm. This property is also known as the self-referential mutation. In other words, while ES, EP, and some other real coded Genetic Algorithms (GAs) require the specification or adaptation of the absolute step size for each variable over iterations, the canonical DE requires

only the specification of a single relative scale factor *F* for all variables. Unlike several other evolutionary computation techniques, basic DE stands out to be a very simple algorithm whose implementation requires only a few lines of code in any standard programming language. In addition, the canonical DE requires very few control parameters (3 to be precise: the scale factor, the crossover rate and the population size) — a feature that makes it easy to use for the practitioners. Nonetheless, DE exhibits remarkable performance while optimizing a wide variety of objective functions in terms of final accuracy, computational speed, and robustness. It is interesting to note that the variants of DE have been securing front ranks in various competitions among EAs organized under the IEEE Congress on Evolutionary Computation (CEC) conference series (for details, please see http://www.ntu.edu.sg/home/epnsugan/index_files/cec-benchmarking.htm). It is evident that no other single search paradigm has been able to secure competitive ranking in nearly all the CEC competitions on single-objective, constrained, dynamic, large-scale, multi-objective, and multi-modal optimization problems.

In order to present the flavor of the huge and multi-faceted literature on DE, in 2010, Neri and Tirronen (2010) reviewed a number of DE-variants for the single-objective optimization problems and also made an experimental comparison of these variants on a set of standard benchmark functions. However, the article did not address issues like adapting DE to complex optimization environments involving multiple and constrained objective functions, noise and uncertainty in the fitness landscape, very large number of search variables, and so on. Also, it did not focus on the most recent engineering applications of DE and the developments in the theoretical analysis of DE. In this respect, the first comprehensive survey on almost all aspects of the DE family of algorithms was published in 2011 by Das and Suganthan (2011). Since then, DE has advanced a lot due to the continuous efforts of EC researchers all over the globe. In a recent survey by Dragoi and Dafinescu (2015), the authors reviewed two aspects of the DE family of algorithms: the self-adaptive and adaptive parameter control strategies in DE and the hybridization of DE with other algorithms. In this article, we present a more exhaustive account of the recent advances in DE including its basic concepts, different structures, and variants for solving constrained, multi-objective, dynamic, and large-scale optimization problems as well as applications of DE variants to practical optimization problems. In addition we present several open research issues that call for the attention of the DE researchers.

The rest of this paper is arranged as follows. In Section 2, the basic concepts related to classical DE are explained along with the original formulation of the algorithm in the real number space. Section 3 discusses the recently developed parameter adaptation and control schemes for DE. Section 4 provides an overview of several prominent variants of the DE algorithm for the single-objective global numerical optimization. Section 5 provides an extensive survey on the applications of DE to the constrained, multi-objective, multi-modal, combinatorial, and dynamic optimization problems. Hybrid DE algorithms have been reviewed in Section 6. An overview on the recent theoretical studies of DE has been presented in Section 7. Section 8 provides an account of the recently developed parallel and distributed DE schemes. Section 9 highlights the recent and prominent engineering applications of DE. Section 10 discusses some interesting future research issues related to DE. Finally, the paper is concluded in Section 11.

2. The Canonical DE Algorithm

The initial iteration of a standard DE algorithm consists of four basic steps – initialization, mutation, recombination or crossover, and selection, of which, only the last three steps are repeated into the subsequent DE iterations. The iterations continue till a termination criterion (such as exhaustion of maximum functional evaluations) is satisfied.

2.1. Initialization of the Decision Variable vectors

DE searches for a global optimum point in a d-dimensional real decision variable space $\Omega \subseteq \mathbb{R}^d$. It begins with a randomly initiated population of *Np* d-dimensional real-valued decision vectors. Each vector, also known as *genome/chromosome*, forms a candidate solution to the multi-dimensional optimization problem. We shall denote subsequent iterations in DE by $t = 0, 1, \ldots, t_{\max}$. Since the parameter vectors are likely to be changed over different iterations, we may adopt the following notation for representing the i^{th} vector of the population at the current iteration:

$$\mathbf{x}_i{}^{(t)} = \left(x_{i,1}{}^{(t)}, x_{i,2}{}^{(t)}, \ldots, x_{i,d}{}^{(t)}\right). \tag{1}$$

For each decision variable of the problem, there may be a certain range within which the value of the decision variable should be restricted, often because decision variables are related to physical components or measures that have natural bounds (for example if one decision variable is a length or mass, we would want it not to be negative). The initial population (at $t = 0$) should cover this range as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum bounds: $\mathbf{x}_{\min} = \left(x_{\min,1}, x_{\min,2}, \ldots, x_{\min,d}\right)$ and $\mathbf{x}_{\max} = \left(x_{\max,1}, x_{\max,2}, \ldots, x_{\max,d}\right)$. Hence, we may initialize the j^{th} component of the i^{th} decision vector as:

$$x_{i,j}{}^{(0)} = x_{\min,j} + rand_{i,j}[0,1]\left(x_{\max,j} - x_{\min,j}\right), \tag{2}$$

where $rand_{i,j}[0,1]$ is a uniformly distributed random number lying between 0 and 1 (actually $0 \leq rand_{i,j}[0,1] \leq 1$) and is instantiated independently for each component of the *i*-th vector.

2.2 Mutation with difference vectors

After initialization, DE creates a *donor/mutant* vector $\mathbf{v}_i{}^{(t)}$ corresponding to each population member or *target* vector $\mathbf{x}_i{}^{(t)}$ in the current iteration through mutation. Five most frequently referred mutation strategies are listed below:

"DE/rand/1": $\mathbf{v}_i{}^{(t)} = \mathbf{x}_{R_1{}^i}{}^{(t)} + F\left(\mathbf{x}_{R_2{}^i}{}^{(t)} - \mathbf{x}_{R_3{}^i}{}^{(t)}\right)$. (3a)

"DE/best/1": $\mathbf{v}_i{}^{(t)} = \mathbf{x}_{best}{}^{(t)} + F\left(\mathbf{x}_{R_1{}^i}{}^{(t)} - \mathbf{x}_{R_2{}^i}{}^{(t)}\right)$. (3b)

"DE/current-to-best/1": $\mathbf{v}_i{}^{(t)} = \mathbf{x}_i{}^{(t)} + F\left(\mathbf{x}_{best}{}^{(t)} - \mathbf{x}_i{}^{(t)}\right) + F\left(\mathbf{x}_{R_1{}^i}{}^{(t)} - \mathbf{x}_{R_2{}^i}{}^{(t)}\right)$. (3c)

"DE/best/2": $\mathbf{v}_i{}^{(t)} = \mathbf{x}_{best}{}^{(t)} + F\left(\mathbf{x}_{R_1{}^i}{}^{(t)} - \mathbf{x}_{R_2{}^i}{}^{(t)}\right) + F\left(\mathbf{x}_{R_3{}^i}{}^{(t)} - \mathbf{x}_{R_4{}^i}{}^{(t)}\right)$. (3d)

"DE/rand/2": $\mathbf{v}_i{}^{(t)} = \mathbf{x}_{R_1{}^i}{}^{(t)} + F\left(\mathbf{x}_{R_2{}^i}{}^{(t)} - \mathbf{x}_{R_3{}^i}{}^{(t)}\right) + F\left(\mathbf{x}_{R_4{}^i}{}^{(t)} - \mathbf{x}_{R_5{}^i}{}^{(t)}\right)$. (3e)

The indices $R_1{}^i, R_2{}^i, R_3{}^i, R_4{}^i$ and $R_5{}^i$ are mutually exclusive integers randomly chosen from the range [1, *Np*], and all are different from the base index *i*. These indices are randomly generated anew for each donor vector. The scaling factor *F* is a positive control parameter for scaling the difference vectors. $\mathbf{x}_{best}{}^{(t)}$ is the best individual vector with the best fitness (*i.e.* with the lowest objective function value for a minimization problem) in the population at iteration *t*. The general convention used for naming the various mutation strategies is DE/x/y/z, where DE stands for Differential Evolution, x represents a string denoting the vector to be perturbed and y is the number of difference vectors considered for perturbation of x. z stands for the type of crossover being used (*exp*: exponential; *bin*: binomial). Note that in the DE/current-to-best/1 scheme the vector which is

being perturbed with the scaled difference of the two other population members is basically a convex combination of the current target vector and the best population member for $F < 1$. This means here the base vector for mutation denotes a point on the line joining the target vector and the best population member and it is an arithmetic recombination between $\mathbf{x}_i{}^{(t)}$ and $\mathbf{x}_{best}{}^{(t)}$. Thus, the resulting donor vector can be thought of as a mutated recombinant.

2.3 Crossover

Through crossover, the donor vector mixes its components with the target vector $\mathbf{x}_i{}^{(t)}$ to form the trial/offspring vector $\mathbf{u}_i{}^{(t)} = \left(u_{i,1}{}^{(t)}, u_{i,2}{}^{(t)}, \dots, u_{i,d}{}^{(t)}\right)$. The DE family of algorithms commonly uses two crossover methods - *exponential* (or two-point modulo) and *binomial* (or uniform) (Das and Suganthan, 2011). We here elaborate the binomial and the exponential crossover schemes. Binomial crossover is performed on each of the d variables whenever a randomly generated number between 0 and 1 is less than or equal to a pre-fixed parameter *Cr*, called the crossover rate. In this case, the number of parameters inherited from the donor has a (nearly) binomial distribution. The scheme can be expressed as:

$$u_{i,j}{}^{(t)} = \begin{cases} v_{i,j}{}^{(t)} & \text{if } j = K \text{ or } rand_{i,j}[0,1] \leq Cr, \\ x_{i,j}{}^{(t)} & \text{otherwise,} \end{cases} \tag{4}$$

where K is any randomly chosen natural number in $\{1,2,\dots,d\}$, $rand_{i,j}[0,1]$ is a uniform random number in [0, 1] and "$j = K$" ensures that $\mathbf{u}_i{}^{(t)}$ gets at least one component from $\mathbf{v}_i{}^{(t)}$. $rand_{i,j}[0,1]$ is instantiated once for every component of each vector per iteration.

In exponential crossover, we first choose an integer n randomly among the numbers $\{1,2,\dots,d\}$. This integer acts as a starting point in the target vector, from where the crossover or exchange of components with the donor vector starts. We also determine another integer L from the numbers in $\{1,2,\dots,d\}$. L denotes the number of components the donor vector actually contributes to the target vector. The integer L is drawn from numbers in $\{1,2,\dots,d\}$ according to the following pseudo-code:

$L = 0$;
DO
{
    $L = L + 1$;
}WHILE (($rand[0,1] < Cr$) AND ($L < d$));

Here also *Cr* is called the *crossover rate* and appears as a control parameter of DE just like *F*. After choosing n and L, the trial vector is obtained as:

$$u_{i,j}{}^{(t)} = \begin{cases} v_{i,j}{}^{(t)} & \text{if } j = \langle n \rangle_d, \langle n+1 \rangle_d, \dots, \langle n+L-1 \rangle_d, \\ x_{i,j}{}^{(t)} & \text{otherwise,} \end{cases} \tag{5}$$

where the angular brackets $\langle . \rangle_d$ denote a modulo function with modulus d *i.e.* $\langle x \rangle_d = x \bmod d$. Hence in effect, probability $(L \geq \upsilon) = (Cr)^{\upsilon-1}$ for any $\upsilon > 0$. For each donor vector, a new set of n and L must be chosen as shown above. The exponential crossover is effective only when linkages exist between the neighboring decision variables (Tanabe *et al*. 2014b). In fact, exponential crossover performed well on 2011 large scale global optimization (LSGO) problems developed for special issue in Soft Computing (Zhao *et al*. 2011, Zhao *et al*. 2013), as many of these problems had linkages among the neighboring decision variables. Due to this limitation of exponential crossover, binomial crossover has been more frequently used.

Although not as common as the binomial or exponential crossovers, some of the DE variants may include the arithmetic recombination (Price *et al*., 2005) where the trial may be generated as a convex combination of the target vector and a donor vector in the following manner:

$$\mathbf{u}_i{}^{(t)} = \mathbf{x}_i{}^{(t)} + k_i\left(\mathbf{v}_i{}^{(t)} - \mathbf{x}_i{}^{(t)}\right), \tag{6}$$

where k_i is a scalar combination coefficient. If k_i remains the same for all the components of the i^{th} vector, then the resulting $\mathbf{u}_i{}^{(t)}$ becomes a point on the line joining solution-points denoted by $\mathbf{x}_i{}^{(t)}$ and $\mathbf{v}_i{}^{(t)}$. In this case, the process is called a line recombination which is rotation invariant. However, the combination coefficient can be randomly instantiated for each component resulting in a component-level operation given by the following at the expense of rotational invariance property:

$$u_{i,j}{}^{(t)} = x_{i,j}{}^{(t)} + k_{i,j}\left(v_{i,j}{}^{(t)} - x_{i,j}{}^{(t)}\right).$$

2.4 Selection

Selection determines whether the target (parent) or the trial (offspring) vector survives to the next iteration *i.e.* at $t = t + 1$. The selection operation is described as:

$$\mathbf{x}_i{}^{(t+1)} = \begin{cases} \mathbf{u}^{(t)} & \text{if } f\left(\mathbf{u}^{(t)}\right) \le f\left(\mathbf{x}_i{}^{(t)}\right), \\ \mathbf{x}_i{}^{(t)} & \text{otherwise,} \end{cases} \tag{7}$$

where $f(.)$ is the objective function to be minimized. Therefore, if the new trial vector yields an equal or lower value of the objective function, it replaces the corresponding target vector in the next iteration; otherwise, the target is retained in the next generation population. The equality in "≤" of (7) helps the DE population to navigate the flat portions of a fitness landscape and to reduce the possibility of population becoming stagnated.

Due to the crossover operation, only the target and trial vector can have the same numerical values for some decision variables. If many population members have the same numerical values for the same decision variables, the difference-vector's components will be zero for these decision variables if such population members are selected for computing the difference vector. However, the parent-offspring competition eliminates the possibility of two distinct population members inheriting the same numerical values for any particular decision variable in the early stages of evolution (Suganthan, 2015).

Note that the DE algorithms can use both synchronous and asynchronous modes of survivor selection or population update. In synchronous population update strategy the population is updated with all the new solutions at the same time, instead of updating the individuals just after being generated. Canonical DE (Storn and Price, 1995; 1997) implements a synchronous update by maintaining a primary array for holding the current individuals and a secondary array to store the selected solutions for the next generation. Once the calculations with the current population members finish at a generation, the secondary array and the primary array are exchanged for resuming calculations in the next generation. In asynchronous population update, each newly generated offspring can replace its parent (if better or equal) and become a member of the current population, subsequently taking part in the mutation of the other population members. Many DE-variants published subsequently (see Das and Suganthan, 2011) have used this update strategy. Asynchronous update permits the improved solutions to contribute to the evolution immediately and can speed up the convergence faster than the synchronous batch mode update.

3. Strategy Selection and Parameter Adaptation in DE

The classical DE includes a set of basic mutation strategies (most common 5 of which have been shown in eqn. (3a) – (3e)) along with three possible crossover schemes (binomial, exponential and arithmetic). Starting

with the landmark paper on Self adaptive DE (SaDE) (Qin *et al*., 2005; 2009), there has been a growing trend of selecting the offspring generation strategies (a combination of a mutation and a crossover strategy) from a pool (Spears, 1995) as well as adapting the control parameters *F* and *Cr* based on success histories of the generated trial vectors. This section provides an overview of such approaches developed after 2010. As will be evident, some of these approaches also create a pool comprising a mixture of the existing and newly proposed mutation schemes.

It is very difficult to classify the DE methods discussed in this section under a well defined taxonomy since some of these approaches combine multiple mechanisms together. We still elaborate these methods under a few representative heads like: DE methods with both strategy and control parameter adaptation, DE with only control parameter (*F* and *Cr*) adaptation, and DE with population size control. Note that although the population size *Np* is counted as a control parameter of classical DE, the volume of works devoted for controlling the population size indicates that this aspect is relatively under investigated. Hence, a separate subsection on this topic is worth including.

3.1 DE with both Strategy and Control Parameter (*F* and *Cr*) Adaptation

DE methods discussed under this subsection have two aspects. Firstly, instead of using a single offspring generation strategy, they select one such strategy from a pool of a few possible strategies. Secondly, they include some mechanisms for adapting *F* and *Cr* values also.

We begin with a very competitive method, called DE with Ensemble of Parameters and mutation Strategies (EPSDE) (Mallipeddi *et al*., 2011) to select the individual-specific strategy from a pool of mutation variants as well as values of *F* and *Cr* from sets of discrete candidate values within certain ranges. During initialization each individual of the population is associated with a value of *F* and *Cr*, and a mutation scheme from the strategy pool. If the generated trial is successful in replacing the target vector, then it also follows the target's strategy in the subsequent generation. Else, in the next generation, the target vector of the same index either picks a new strategy or keeps its old one with equal probability. The mutation pool is created with three strategies having distinct characteristics. These strategies are DE/best/2/bin, DE/rand/2/bin, and DE/current-to-rand/1 (chosen for its diverse nature). Note that unlike the first two strategies, DE/current-to-rand/1 does not involve a binomial crossover. Rather, it forms a kind of mutated recombinant (which is rotation invariant) through the following equation:

$$\boldsymbol{u}_i{}^{(t)} = \boldsymbol{x}_i{}^{(t)} + K.\left(\boldsymbol{x}_{r_1}{}^{(t)} - \boldsymbol{x}_i{}^{(t)}\right) + \boldsymbol{F}.\left(\boldsymbol{x}_{r_1}{}^{(t)} - \boldsymbol{x}_{r_2}{}^{(t)}\right),$$

where *K* is an additional combination coefficient. The *F* values are taken in the range [0.4, 0.9] with a step of 0.1, and the *Cr* values are taken in the range [0.1, 0.9] with the same step size. EPSDE exhibited highly competitive performance on the IEEE CEC 2005 benchmark suite for real parameter optimization. EPSDE has since then been enhanced by integrating SaDE-based adaptation (Qin *et al*., 2005 and 2009). Recently, Mallipeddi (2013) proposed a variant of EPSDE where the ensemble of *F* and *Cr* values are evolved by using the optimization process of another metaheuristic algorithm called Harmony Search (HS). However, this algorithm can be computationally costlier, especially on large scale problem instances, due to the involvement of two global optimizers: DE and HS.

Wang *et al*. (2011b) developed a Composite DE (CoDE) algorithm where a trial vector is chosen from a set of candidates generated by using different DE strategies. The authors selected 3 trial vector generation algorithms (DE/rand/1/bin, DE/rand/2/bin and DE/current-to-rand/1) and three popular choices of the control parameter settings ($F = 1.0, Cr = 0.1$; $F = 1.0, Cr = 0.9$; $F = 0.8, Cr = 0.2$). In each generation three candidates for

each trial vector are generated by a distinct strategy which will use a control parameter settings picked uniformly at random from the set of allowable values. The final trial vector will be selected as the candidate with the best fitness value. The three parameter settings used in the work have distinct effects and thus, will generate candidates of different characteristics. The varying *Cr* and *F* values are responsible for different search behaviors of the algorithm. For example, $F = 1.0, Cr = 0.9$ provided high variation in donor and high degree of perturbation of parent (most components of the trial vector coming from the donor), resulting in exploration of the search space, while $F = 0.8, Cr = 0.2$ does the opposite to facilitate exploitation of the space around the target (parent) vector.

Gong *et al*. (2011a) proposed an adaptive DE algorithm where 4 mutation strategies are used to create a pool. These strategies are DE/current-to-*p*best/1, DE/rand-to-*p*best/1 and their modified versions with external archive (Zhang and Sanderson 2009b). For each individual, a particular mutation strategy is selected from the pool with a controlled randomness. The values for the control parameters are also adapted following the JADE algorithm (Zhang and Sanderson 2009b). The same year, Gong and his colleagues (Gong *et al*., 2011b) presented an adaptive strategy selection technique for generating better trial vectors by DE. Besides choosing a suitable strategy pool, the authors also tackled the two issues of selecting a strategy and rewarding the successful one. For each of *k* strategies of the pool, one is selected based on a probability. The probability is improved with the rewards obtained by a strategy. For updating the probabilities, two methods are suggested: the probability matching defined by Goldberg (1990) and the adaptive pursuit defined by Thierens (2005). For assignment of credit the relative fitness improvement measure proposed in Ong and Keane (2004) was used.

Elsayed *et al*. (2011) presented a variant of DE, which adapts a mutation strategy, by selecting one from a pool of four allowable schemes, with the progress of generations. The authors pointed that, different mutation schemes due to their distinct characteristics, cannot perform well in the entire run of the DE algorithm, and proposed that a mutation scheme should be used in those generations only where it will be most beneficial. This proposal is implemented, by dividing the total population, into four distinct groups, each of which will execute one type of mutation strategy on their members. In contrast to the most commonly used mutation schemes of DE, the authors used the following 4 mutation schemes:

"DE/rand/3" : $\boldsymbol{v}_i{}^{(t)} = \boldsymbol{x}_{r_1}{}^{(t)} + F.\left(\boldsymbol{x}_{r_2}{}^{(t)} - \boldsymbol{x}_{r_3}{}^{(t)} + \boldsymbol{x}_{r_4}{}^{(t)} - \boldsymbol{x}_{r_5}{}^{(t)} + \boldsymbol{x}_{r_6}{}^{(t)} - \boldsymbol{x}_{r_7}{}^{(t)}\right)$.

"DE/best/3" : $\boldsymbol{v}_i{}^{(t)} = \boldsymbol{x}_{best}{}^{(t)} + F.\left(\boldsymbol{x}_{r_2}{}^{(t)} - \boldsymbol{x}_{r_3}{}^{(t)} + \boldsymbol{x}_{r_4}{}^{(t)} - \boldsymbol{x}_{r_5}{}^{(t)} + \boldsymbol{x}_{r_6}{}^{(t)} - \boldsymbol{x}_{r_7}{}^{(t)}\right)$.

"DE/rand-to-current/2": $\boldsymbol{v}_i{}^{(t)} = \boldsymbol{x}_{r_1}{}^{(t)} + F.\left(\boldsymbol{x}_{r_2}{}^{(t)} - \boldsymbol{x}_i{}^{(t)} + \boldsymbol{x}_{r_3}{}^{(t)} - \boldsymbol{x}_{r_4}{}^{(t)}\right)$.

"DE/rand-to-best and current/2": $\boldsymbol{v}_i{}^{(t)} = \boldsymbol{x}_{r_1}{}^{(t)} + F.\left(\boldsymbol{x}_{best}{}^{(t)} - \boldsymbol{x}_{r_2}{}^{(t)} + \boldsymbol{x}_{r_3}{}^{(t)} - \boldsymbol{x}_i{}^{(t)}\right)$.

Wu *et al.* (2015a) proposed a multi-population based framework to realize an adapted ensemble of three mutation strategies (i.e. "current-to-pbest/1" and "current-to-rand/1" and "rand/1") into a novel DE variant, named MPEDE. The population of MPEDE is dynamically partitioned into several subpopulations including three indicator subpopulations and one reward subpopulation. Each indicator subpopulation with a relatively smaller size is assigned to a constituent mutation strategy and the reward subpopulation with a relatively larger size is assigned to the currently best performed mutation strategy as an extra reward. This way, dynamically computational resource allocation among the mutation strategies is realized and the best mutation strategy is expected to obtain the most computational resources.

3.2 DE with Adaptation of *F* and *Cr* Parameters Only

Since 2010, there has been a good volume of studies focusing on guided adaptation of the two primary control parameters of DE namely *F* and *Cr*. Instead of some prior works which mainly were based on randomization of

these parameters without any learning mechanism (see for example Das *et al*., 2005), most of these recent studies try to involve some form of learning from the past instances of successes/failures while adapting *F* and *Cr*.

Elsayed *et al*. (2013) presented a variant of DE with an adaptive parameter control scheme. The authors defined two sets of allowable values for *F* and *Cr*, and the trial vector is generated by DE/rand/1/bin strategy (two vectors are selected at random from the population, while the base is selected from the solutions in the range of top 10% to 40% performing individuals similar to JADE strategy (Zhang & Sanderson 2009a)). Initially a counter is assigned and reset to zero for each of the combinations of *F* and *Cr* values from the set. A trial vector is generated by picking a combination of the control parameters randomly, and if it is successfully selected for the next generation, the counter corresponding to the combination is increased by one. After each interval of generations the top half of the performing combinations is selected, their counters are reset to zero, while the others are discarded. If the number of generation exceeds a limit, then the iteration count and combination counters are reset to zero, indicating a restart.

Sarker *et al*. (2014) improved the work of Elsayed *et al*. (2013) to dynamically identify and use the best combination of the control parameters of DE *i.e. Np*, *F* and *Cr*. The algorithm decreases the value of Np after a periodic interval of generations. The better solutions are kept in the reduced population while the others are archived. When least value of Np is reached, the average fitness improvement by the populations having different number of members are calculated, and the Np corresponding to the best improved population is selected as the current population size. The population is increased by adding the top required number of individuals from the archive. The mutation scheme remains the same as that used by Elsayed *et al*. (2013).

Tanabe and Fukunaga (2013) proposed an improved variant of the JADE algorithm (Zhang and Sanderson, 2009a) and called the same as the Success History based DE (SHADE). JADE is a powerful DE - variant that uses a control parameter adaptation strategy based on updating the parameters of the probability distributions from which values of *F* and *Cr* are sampled. Such updating considers those *F* and *Cr* values that yielded successful offspring in the recent generations. In SHADE, instead of sampling the *F* and *Cr* values from gradually adapted probability distributions, the authors used historical memory archives M_{Cr} and M_F which store a set of *Cr* and *F* values, respectively that have performed well in the recent past. The algorithm generates new *Cr*, *F* pairs by directly sampling the parameter space close to one of the stored pairs. Out of the 21 algorithms that participated in the IEEE CEC competition on real parameter single-objective optimization (Liang *et al*., 2013a), SHADE ranked 3rd, the first two ranks being taken by non-DE based algorithms. Tanabe and Fukunaga (2014) further improved the SHADE algorithm by using the linear population size reduction and called this variant as L-SHADE. In L-SHADE, the population size of DE is continually reduced by means of a linear function. If the minimum population size is Np_{min} with the initial being Np_{init}, then the population size in the t^{th} generation is:

$$Np^{(t)} = \left\lfloor \frac{Np_{init} - Np_{min}}{MAX\_NFE} . NFE + Np_{init} \right\rfloor .$$

Here, *NFE* is the current number of fitness evaluations, and *MAX_NFE* is the maximum allowable number of fitness evaluations. As the population size will reduce over generations, only top $Np^{(t)} - Np^{(t+1)}$ individuals will be copied to the next generation. The size of the archive of solutions will also be readjusted with the current population size, by discarding the required number of worse solutions. The only challenge of the algorithm is the requirement of maintaining three archives for solutions and control parameters, which increases the demand

for space. L-SHADE exhibited the best competitive performance among non-hybrid algorithms at the CEC 2014 competition on real parameter single-objective optimization (Liang *et al*., 2013b).

Yu *et al*. (2014a) proposed an individual dependent control parameter adaptation mechanism by using a two steps process. In the first step, the nature of the population is estimated, *i.e.* if the population is in explorative or exploitative state. Depending on the population state then, the values for *F* and *Cr* are adapted. In the second step, the fitness information and the distance of an individual from the best individual is used to modify the population settings to adapt it for that individual. The state of the population can be estimated from the distribution of the population which will be diverse in explorative but clustered in exploitative state. For this, the individuals of a generation are first sorted according to their fitness values to keep the best on top. Then all the individuals are sorted according to their distance for the best fit solution, keeping the nearest on the top. If for the i^{th} individual the fitness based rank is f_i and the distance based rank is d_i, then the population state is quantified by indicator of the optimization state or *IOS* as $IOS = \sum_{i=1}^{Np} |f_i - d_i|$. The *IOS* values are normalized to make them in the range [0, 1]. A low value of the *IOS* signifies that the better solutions are also at a close proximity of the best or the population is converging, while a greater *IOS* is indicating a scattering of better solutions far from the best, or the population is diverse and exploring. Thus, for choosing the state of the population a random number is picked uniformly in the range [0, 1], if it is less than *IOS* then the state of the population is explorative else it is exploitative. Now, in the case of selecting the parameters, an explorative population will demand a high *F* and a high *Cr*, while an exploitative will require the opposite. The *F* and *Cr* values for the population of t^{th} generation are assigned in the following way:

$$F_p^t = \begin{cases} F_p^{t-1} + c_F \Delta F_p \text{ if population is explorative,} \\ F_p^{t-1} - c_F \Delta F_p \text{ if population is exploitative.} \end{cases}$$

$$Cr_p^t = \begin{cases} Cr_p^{t-1} + c_{Cr} \Delta Cr_p \text{ if population is explorative,} \\ Cr_p^{t-1} - c_{Cr} \Delta Cr_p \text{ if population is exploitative.} \end{cases}$$

$$\Delta F_p, \Delta Cr_p = \begin{cases} \dfrac{IOS - IOS_{min}}{IOS_{max} - IOS_{min}} \text{if population is explorative,} \\ \dfrac{IOS_{max} - IOS}{IOS_{max} - IOS_{min}} \text{if population is exploitative.} \end{cases}$$

$$IOS_{min} = 0;\ IOS_{max} = \begin{cases} \dfrac{Np^2}{2} \text{if } Np \text{ is even,} \\ \dfrac{Np^2 - 1}{2} \text{if } Np \text{ is odd.} \end{cases}$$

Now for each of the individuals, the population parameters will be adjusted, and for the i^{th} individual, it will be defined as:

$$F_i^g = \begin{cases} F_p^g + \Delta F_i \text{ if} f_i > {}^{Np}/_2 \text{ and } d_i > {}^{Np}/_{2,} \\ F_p^g - \Delta F_i \text{ if} f_i < {}^{Np}/_2 \text{ and } d_i < {}^{Np}/_{2}, \\ F_p^g \text{ otherwise.} \end{cases}$$

$$Cr_i^g = \begin{cases} Cr_p^g + \Delta Cr_i \text{ if} f_i > {}^{Np}/_2 \text{ and } d_i > {}^{Np}/_{2}, \\ Cr_p^g - \Delta Cr_i \text{ if} f_i < {}^{Np}/_2 \text{ and } d_i < {}^{Np}/_{2}, \\ Cr_p^g \text{ otherwise.} \end{cases}$$

$$\Delta F_i, \Delta Cr_i = \begin{cases} \dfrac{f_i + d_i - Np}{2Np} \text{if} f_i > {}^{Np}/_2 \text{ and } d_i > {}^{Np}/_{2}\ , \\ \dfrac{Np - f_i - d_i}{2Np} \text{if } f_i < {}^{Np}/_2 \text{ and } d_i < {}^{Np}/_{2}. \end{cases}$$

Value of the coefficients c_F and c_{Cr} in the expressions for F_i^g and Cr_i^g, were set to 0.1 and 0.05, respectively by the authors based on empirical observations. Although the algorithm reportedly yields competitive results, the adaptation mechanisms incur several auxiliary calculations and distance computations which can be expensive for complex and high-dimensional functional landscapes.

To non-linearly change both the values of *F* and *Cr* along with their direction of changing with the progress of iterations, Draa *et al*. (2015) proposed six adaptation schemes based on sinusoidal functions. In these schemes, the *F* and *Cr* values can be obtained from scaled and shifted sinusoidal terms. In some of these schemes, either *F* or *Cr* values are also fixed to constants (in particular *F* to 0.5 and *Cr* to 0.9). The scaling factors for the sinusoidal terms are either increased or decreased linearly with iteration number. It is experimentally shown that scheme 2, in which both *F* and *Cr* values are sampled from sinusoidal terms with increasing coefficients, outperforms other proposed schemes. However, the technique used a new parameter *i.e.* the frequency of the sinusoidal function, the value of which can be estimated from empirical study, provided by the authors.

In the context of distributed DE algorithms, Falco *et al*. (2014) addressed the popular problem of selecting the control parameters and updating them, by introducing three new strategies. The initial values of *F* and *Cr* are selected uniformly at random from the range [0.1, 1]. The migration among distributed DE sub-populations takes place in every say *T* generations, and then at the end of each such interval, the average fitness improvement (over the interval of generations) is calculated for each sub-population. One of the sub-populations is randomly chosen following one of the three control parameter updating strategies (elaborated next), whose *F* and *Cr* values are randomly selected from the above mentioned interval. In the first updating scheme, one of the sub-populations is chosen from only those, for which the average fitness improvement is less than the average of the average fitness improvement of the sub-populations. In the second scheme the sub-population is picked at random from any of the sub-populations except the one containing the current global best. In the third the sub-population holding the current global worst is selected for replacement of the control parameters. The migration policy is simple; each sub-population selects all those solutions having better fitness than the average fitness, and sends them to all of its neighbors. After receiving all the solutions from its neighbors, a subpopulation forms an intermediate set with its native solutions and the newcomers. A subpopulation then selects the required number of top solutions according to its cardinality. While adapting *F* and *Cr*, the algorithm also introduces a new parameter, which is the interval of generation (*T*). Mallipeddi *et al*. (2014) proposed a scheme to adapt the *F* and *Cr* parameters in DE by using a Gaussian adaptation algorithm. Gaussian adaptation, which is a stochastic process of adapting a Gaussian distribution to a region or points in some feasible search space, is employed to determine an optimal combination of *F* and *Cr*. However, as is evident from the results reported by the authors, the algorithmic complexity can increase with the increase of dimensions for this algorithm.

In a recent work, Tang *et al*. (2015) suggested that for each individual a clue about the most suitable trial vector generation strategy can be obtained by observing its fitness value. For example a solution with low fitness (meaning high objective function value for a minimization problem) needs a higher *F* value to explore the search space, while the one with high fitness needs a small *F* value to exploit its neighborhood. Similarly a less fit parent needs to be perturbed more and will be helped by a high *Cr* value, than a fitter parent. Two schemes were presented for the parameter selection. In the first scheme the population is sorted based on the ascending fitness values, *F* and *Cr* for any individual are determined as the ratio of its rank and *Np*. The second scheme is based on the individual fitness value, for any i^{th} individual the corresponding parameters are calculated as:

$$F_i \text{ or } Cr_i = \frac{f_i - f_L + \delta_L}{f_U - f_L + \delta_L}.$$

Here f_U and f_L are the maximum and minimum fitness values of the current population respectively and δ_L is the difference between minimum and second minimum fitness values in the current generation. However, such strict parameter settings can take away the benefits of randomization and hence the parameters are actually sampled from a random normal distribution with mean as the calculated parameter value and standard deviation 0.1. For the individual mutation strategy, the concept of base vector is defined say $\mathbf{x}_b$, which may differ from the target vector, and the value of F for mutation will be taken for the base vector. From experiments, it is justified that using the base vector as the same as the trial vector helps in the early generations while a random vector in the late generations helps the convergence process. To distinguish between early and late generations, a generation threshold is defined, which is problem dependent. The authors suggested if the ratio of the successful number of selected offspring for the next generation and *Np* reaches under a threshold close to zero, and retains that state for a predefined number of consecutive iteration then the early generation stage can presumed to be ended. To further elevate the scheme the authors also suggested to use different mutation schemes for differently fit individuals. The population is divided into two non-overlapping sets: *superior* and *inferior*. Depending on the generation and an individual's fitness; its mutation scheme will be selected from any one of the current-to-rand/1, current-to-better/1, rand/2 and rand-to-better/1. To facilitate a gradual convergence, the size of the superior set will be increased with the generation, which is defined for the i^{th} generation to be $Np * (0.1 + 0.9^{10^5 t/(t_{max}-1)}$. Finally the mutation scheme for the i^{th} vector is defined as follows, where $\boldsymbol{x}_b$ is the base vector.

$$\mathbf{v} = \begin{cases} \mathbf{x}_b + F_b(\mathbf{x}_{r_1} - \mathbf{x}_b) + F_b(\mathbf{x}_{r_2} - \mathbf{d}_{r_3}) \; when \; i \in superior, \\ \mathbf{x}_b + F_b(\mathbf{x}_{better} - \mathbf{x}_b) + F_b(\mathbf{x}_{r_2} - \mathbf{d}_{r_3}) \; when \; i \in inferior. \end{cases}$$

Here, $\mathbf{x}_b = \mathbf{x}_i$ in the earlier stage of generation else $\mathbf{x}_b$ is selected at random from [1, *Np*]. The vector $\mathbf{x}_{better}$ is randomly selected from the set *superior*. Each component of the vector $\mathbf{d}_{r_3}$ is either uniformly randomly sampled from the entire feasible search space with a probability p_d (the authors call it "the probability factor of disturbance") or it is copied from the original population member of the same index *i.e.* from $\mathbf{x}_{r_3}$. This procedure can suffer from the need of heuristic tuning of the other parameters like p_d for real world problems.

Instead of resorting to complicated parameter adaptation schemes or incorporating additional local search methods recently Das *et al*. (2015) proposed a simple DE strategy to solve large scale optimization problems where the values of the *F* and *Cr* are switched in a uniformly random way between two extreme corners of their feasible ranges for different population members. Also each population member is mutated either by using the DE/rand/1 scheme or by using the DE/best/1 scheme. The population member is subjected to that mutation strategy which was responsible for the last successful update at the same population index under consideration. This scheme exhibited very competitive performance on the high-dimensional functions compiled under the CEC 2008 and 2010 competitions on large scale global optimization. The results indicate that a combination of the high and unconventional value of *F* (= 2) with the low value (= 0.5) can be indeed very useful for solving benchmark functions. In contrast to the usual DE algorithms that sample *F* values from the interval of (0.4, 1) and *Cr* values from (0, 1), the work of Das *et al*. (2015) indicate that most of the useful information about *F* and *Cr* values can remain attached to the boundaries of their feasible regions. This point requires further analytical and experimental investigations in future.

Controlling the degree of randomization of *F* and *Cr* and investigating the effect of the same on the performance of DE are important research directions in DE. A very interesting work in this direction was recently published by Zamuda and Brest (2015). Here, the authors introduced a randomness level parameter,

which influences the dynamics of the control parameter values and their propagation through "suitable individuals' improvement contributions during elitist selection" in an extended framework of the jDE algorithm (Brest *et al*., 2006). Different randomization frequency parameter values (from default at 10% evaluations of jDE), rendered the performance of the utilized SPSRDEMMS (Structured Population Size Reduction DE with Multiple Mutation Strategies) algorithm (Zamuda *et al*., 2013) on top or bottom of a ranking list comprised of algorithms from the CEC 2013 real parameter optimization competition. Relation between adaptation and self-adaptation impact was shown, displaying when and how, different number of significant improvements within the algorithm occur and that different frequencies are suitable for different optimization problems, also differing for *F* and *Cr* randomization.

A few more interesting schemes for strategy and control parameter adaptation of DE are summarized in Table 1.

Table 1: Summary of some DE schemes with strategy and/or control parameter adaptation mechanisms

Scheme	Authors	Strategy Selection Mechanism	Parameter Adaptation Mechanism
Multi Population DE algorithm (MPDE)	(Yu and Zhang, 2011)	The population is divided into a number of subpopulations. The subpopulation properties are kept fixed throughout the entire run of the algorithm. The migration of information is done using a DE/best/1 type mutation where the base is the local best of the current subpopulation, while the two random vectors can be taken from the entire population.	Inspired by the work of Zhan and Zhang (2010), each subpopulation maintains their own set of parameter values. After each generation the parameter values of the subpopulation generating maximum number of successful trials will be used to update the parameter values of all the subpopulations.
Self-adaptive Mutation DE (SaMDE)	(Silva *et al*., 2011)	The individual specific mutation scheme is selected from a pool (containing DE/rand/1, DE/best/1, DE/best/2 and DE/current-to-rand/1) by a roulette wheel strategy. The probability for choosing a mutation scheme is updated by a DE/rand/1 mutation operation on the probability vector (the scale factor is randomly selected from the range $[0.7, 1]$) after each generation.	The value of F and Cr is updated by applying the selected mutation strategy (the scale factor is randomly selected from the range $[0.7, 1]$) on the current values.
Fitness Adaptive DE (FiADE)	(Ghosh *et al*. 2011)	Only DE/best/1/bin is used for trial generation purpose.	Two schemes for obtaining an individual specific value of F are suggested. Both of them are guided by the difference of fitness between the individual and the current best of the population. The value of Cr is also adapted based on the fitness of the donor vector compared to the current best.
Modified DE (MDE) with *p*-best crossover (MDE-*p*BX)	(Islam *et al*., 2012)	A new mutation scheme was proposed.	For *F*, a Cauchy distribution with an adaptive location parameter (F_l) and a scale of 0.1 is used. The new F_l is calculated as an weighted average (the weight is greater for the current F_l and randomly taken from the range $[0.8, 1]$) of the current F_l and the power mean of the archive of the better performing F_l values till the current generation.. The Cr is calculated with a similar technique only a Gaussian distribution is used, with an adaptive mean and standard deviation 0.1.
Teaching and Learning Based Self-adaptive DE (TLBSaDE)	(Biswas *et al*., 2013)	A pool of four mutation strategies (DE/rand/1, DE/rand-to-best/2, DE/rand/2, and DE/current-to-rand/1) is kept, and one of them is selected by a roulette wheel. The probability of selecting a strategy is calculated from the success rates of all the available strategies over the last LP generations.	The scaling factor is sampled from the distribution $N(0.5, 0.3)$ and crossover rate is picked from the distribution $N(CR_m, 0.1)$. The CR_m is calculated as the mean of successful CR values over the last LP generations.
Modified DE algorithm (MDE)	(Zou *et al*. , 2013)	One of the two mutation strategies, DE/rand/1 and DE/best/1, is selected for each individual based on a probability. The probability will favor DE/best/1 with the progress of the run of the algorithm.	The scale factor is selected from a Gaussian distribution with fixed mean and standard deviation. The crossover rate is randomly picked from a predefined range.

Adaptive DE	(Bujok *et al*., 2014)	Adaptively selects a trial vector generation scheme from a pool of six DE strategies using the *randrl* mutation (Tvrdík, 2009) along with the DE/current-to-*p*best/1 mutation with orthogonal crossover. A strategy is selected by a roulette wheel where the probabilities are calculated based on the success rate of the strategies.	The scale factor is 0.8 for all the strategies using DE/rand/1 mutation, and 0.5 for the one using DE/current-to-*p*best/1 mutation. The crossover rate is different but kept as a constant for all of the strategies.
DE with crossover rate repair.	(Gong *et al*., 2014)	The JADE mutation scheme is directly used.	The *Cr* value is drawn from the distribution in the same way as JADE. However, the binomial crossover is rewritten in the form $\mathbf{u}_i = B_i\mathbf{v}_i + (1 - B_i)\mathbf{x}_i$. Here B_i is the i^{th} component of a binary vector $\boldsymbol{B}$ used for generating the i^{th} trial. Clearly $\mathbf{u}_i$ can be either equal to $\mathbf{v}_i$ or to $\mathbf{x}_i$ depending on whether B_i is 1 or 0. The Cr value will be updated (after generating trail) as the mean of the elements of $\boldsymbol{B}$, and will be kept in the archive.

3.3 DE with Adaptive Population Size Control

Compared to the huge body of works on the adaptation of *F*, *Cr* and the offspring generation strategy, relatively fewer works have been undertaken in recent past to control or adapt the population size in DE.

Brest and Maučec (2011) proposed a new population reduction technique in conjunction with 3 parameter adaptation strategies. These strategies use the basic jDE (Brest *et al*., 2006) type randomization of *F* and *Cr* and are denoted by the authors as jDEbin (self-adaptive DE/rand/1/bin), jDEexp (self-adaptive DE/best/1/bin) and jDEbest (self-adaptive DE/best/1/bin). In each iteration of the algorithm, one of these 3 strategies remains active. The population reduction mechanism amounts to reduce, according to a prearranged schedule, the population size of DE. More specifically, the computational budged is divided into periods. At the beginning of each period, the population size is halved. In the proposed population reduction mechanism, each individual is allowed to compete with another individual belonging to the same offspring generation strategy as the former one. In this way, the DE appears to progressively focus the search and thus prevents the undesired stagnation effect. A similar scheme of progressive reduction of the population size keeping the total computational budget (in terms of maximum number of function evaluations allowed) constant was discussed in context to compact DE (to be discussed in Section 4) by Iacca *et al*. (2011). Zamuda and Brest (2012) reduced the population size of jDE with increasing number of FEs and used two different mutation strategies depending on the population size. This algorithm provided competitive results on the real life problems compiled under the 2011 CEC competition on testing EAs on practical optimization problems. Zamuda *et al*. (2013) combined the population reduction based jDE with two mutation schemes by using a structured population and the resulting SPSRDEMMS algorithm was tested on the CEC 2013 benchmarks for real parameter optimization (Liang *et al*., 2013a).

Instead of monotonically reducing the population size, Yang *et al*. (2013) proposed a method to adapt the same depending on the population diversity. Their method is capable of identifying the instance when the population lacks enough diversity, measured in terms of the pair-wise Euclidean distance among the individuals. When the moment is identified, the population is regenerated to enhance the diversity, thus eliminating the chances of stagnation. However, for high-dimensional problems this method can be fairly expensive due to the need for computing Euclidean distances several times.

Zhu *et al*. (2013) proposed a method that varies the population size in a specified range. The algorithm after an iteration of DE, detects the status and takes the decision of decreasing or increasing the current population size, and performs accordingly. If the DE cannot find a better solution in successive generations, then it is stagnating

and the population needs to be expanded by introducing new solutions. If the DE updates the best individual in successive iterations, then it is filling the population with redundant solutions, and the population size is decreased to purge the redundancy. If the population size reaches a bound and retains it for a number of generations, then the population size is decreased or increased depending on the bound it reached. To downsize the population, the solutions are first sorted (keeping the best on the top). The rank of the i^{th} solution is determined as follows.

$$rank_i = \left\lfloor f_i - f_{min} \Big/ \frac{f_{max} - f_{min}}{\omega} \right\rfloor,$$

where $\omega = \lfloor 0.8 * populationSize \rfloor$. For each of the individuals, a probability is calculated based on their rank and it is selected for purging based on that probability. From all the individuals selected for purging, the required numbers of solutions are randomly deleted by the algorithm. At the time of increasing, a set of elite members are selected and new solutions are generated by perturbing them, in a selected number of dimensions. However, in the process of making the population size adaptive, the algorithm introduces a number of new control parameters to monitor the status or to generate new individuals.

The work of Mallipeddi *et al*. (2014) was further extended by Gonuguntla *et al*. (2015) where besides the Gaussian adaptation technique, in each generation, a fixed number of individuals are sampled from the large set to be included in the population of the DE algorithm. This fixed number of individuals can be selected from the large set either randomly or based on the objective value depending on the stage of evolution.

4. Prominent DE Variants for Bound-constrained Single-objective Global Optimization

Since its inception, DE has been most frequently applied to the global optimization problems involving a single objective function and bound constraints on the decision variables. After 2010, most of the single-objective DE variants use parameter adaptation schemes and multiple mutation and crossover strategies. Hence, the demarcation between algorithms described in this section and those discussed in the previous section may not be that prominent. However, in this section we confine ourselves to the review of those DE methods that use significant heuristic mechanisms to improve their search moves, besides strategy selection and parameter adaptation techniques. We review the DE algorithms that have been published in front-ranking journals and conferences that cover the evolutionary computation area. We discuss such DE variants under the following seven heads although such grouping is not very rigid always, as a proposal may use a combination of techniques from different heads. These heads are DE with new initialization techniques, DE with new or improved mutation techniques, DE with new or improved crossover techniques, DE with sub-population and cluster based improvisations, population topology and population diversity guided DE, DE with representation of individuals in probabilistic space, and DE with parent selection framework. Nevertheless, some worth mentionable DE methods that cannot be fitted to these seven heads are discussed under “Other Techniques” in Section 4.8.

4.1 DE with New Initialization Techniques

An intelligent selection of vectors from promising regions of the feasible search space to initialize the population will definitely encourage DE to converge faster and obtain better solution. Melo and Delbem (2012) proposed such a technique called Smart Sampling (*SS*) to identify promising regions of the search space where an optimum may lie. At first, a high number of random solutions are generated such that the search space can be covered. Based on the fitness value this initial population is filtered and only better solutions are kept. A

classifier is trained to distinguish the promising solutions from the non promising ones. A collection of new solutions are generated by moving one of the population members towards one of the better solutions, with a random noise. Now the classifier is used to identify the good solutions from the newly introduced collection. The good individuals are added to the population, and the increased population is downsized to maintain a fixed cardinality by deleting the worse members. This process is repeated until a convergence criterion is met. After the iterations a rule based classifier is used on the final population to identify the promising region. DE is initialized with members from this promising region alongside a small percentage of random vectors in the search space. While the procedure can be useful, it is highly expensive to apply, especially for multimodal and high dimensional problems as noted by the authors.

Poikolainen *et al*. (2015) proposed a cluster-based population initialization technique for DE. The initialization is performed as a three staged process. In the first phase, two local searches are performed on a random collection of uniformly selected points over the search space. The first local search translates each vector with a certain amount say ς. If the translated vector is better than the original, then it replaces the original vector in the population. If the original vector is fitter than the modified, then another modification is done to the original by translating it in the opposite direction of the previous alteration, with an amount of $\varsigma/2$. If the new modified vector is fitter than the original, then the population is updated. If none of the vectors of the population is updated, then the ς valued is halved, and the process iterates until terminal condition is met. The second searcher is the Rosenbrock algorithm (Rosenbrock, 1960), which has been shown to converge towards a local optimum. In the second phase *k*-means clustering is applied to the resultant population of the first phase. To choose the value of *k* i.e. the number of clusters, the silhouette index is used (Rousseuw, 1987). After the *k*-means clustering phase, the population may not contain *Np* number of individuals. The third stage is used to generate additional individuals to form the initial population of DE. In the third stage the best fit individuals are collected (forming a set *Q*) from each of the cluster. A probability is assigned to each of these fitter individuals based on their fitness values. Now a member of *Q* is selected by a roulette wheel based on the associated probability, and a new individual is generated by a Gaussian distribution with the selected vector as the mean and a specified standard deviation.

4.2 DE with New or Improved Mutation Strategies

Since 2010, apart from the combination of the existing mutation schemes in DE, researchers have also made several attempts to devise new mutation schemes (involving difference vectors in various forms) which can provide improved search moves on complex fitness landscapes. Often some improvements of the existing mutation strategies based on neighborhood information or some other heuristics (like 2-opt algorithm) have also been reported in literature. A few prominent approaches in this direction published in front ranking journals are discussed below.

4.2.1 2-Opt based DE (2-Opt DE)

To achieve a faster and better convergence than classical DE, Chiang *et al*. (2010) proposed a variant of DE, where a mutation scheme influenced by the 2-Opt algorithm (Croes, 1958) is used instead of the regular DE/rand/1 or DE/rand/2 schemes. 2-Opt algorithm was originally designed for finding routes in the travelling salesperson problem, and was later applied to genetic algorithm, simulated annealing, etc. The original 2-Opt local search prohibits routes from self-crossing by reordering them. This provides a good chance of escaping from local optima. Following the essence of this idea, the authors devise a DE/2-Opt/1 mutation scheme involving three vectors with indices r_1, r_2 and r_3 from the current population, and corresponding to the i^{th} target vector as follows (for minimization problems):

$$\mathbf{v}_i = \begin{cases} \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3})\ if\ f(\mathbf{x}_{r_1}) < f(\mathbf{x}_{r_2}), \\ \mathbf{x}_{r_2} + F(\mathbf{x}_{r_1} - \mathbf{x}_{r_3})\ otherwise. \end{cases}$$

Thus, in 2-opt/1 scheme the base vector always has better (or equal) fitness as compared to the first member forming the difference vector. For a 2-Opt/2 version the mutation is given as follows for the i^{th} target vector:

$$\mathbf{v}_i = \begin{cases} \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) + F(\mathbf{x}_{r_4} - \mathbf{x}_{r_5})\ if\ f(\mathbf{x}_{r_1}) < f(\mathbf{x}_{r_2}), \\ \mathbf{x}_{r_2} + F(\mathbf{x}_{r_1} - \mathbf{x}_{r_3}) + F(\mathbf{x}_{r_4} - \mathbf{x}_{r_5}) otherwise. \end{cases}$$

Here, the indices r_1, r_2, r_3, r_4 and r_5 are mutually distinct and picked randomly from the population, similar to the DE/rand/2 strategy of (3e). However, apart from experimental results, the authors provided no intuitive justification for the improved performance of these schemes over the conventional DE mutations.

4.2.2 Proximity-based DE (ProDE)

Epitropakis *et al*. (2011a) proposed a proximity induced mutation scheme for DE, where neighbors of a parent vector, rather than the random ones will be used to generate the donor vector. To avoid sacrificing exploration capability, a probabilistic approach was suggested and empirical results demonstrating mutation dynamics of a DE run was provided to justify the approach. The mutation scheme first computes the pair-wise distance between all members of a population, and stores them in a matrix say $R = [r_{ij}]_{Np*Np}$, where r_{ij} is the distance between i^{th} and j^{th} member of the population. From these pair-wise distances a pair-wise probability is calculated as follows, where the minimum distant neighbor of a vector will have the highest probability to be selected as r_i index. Let us take the probability matrix to be R_p. Then,

$$R_p(i,j) = 1 - \frac{r_{ij}}{\sum_{k=1}^{Np} r_{ik}}.$$

Now for each index i, select three vectors with indices $k = r_1$, r_2, and r_3 from the population (where $r_1, r_2, r_3 \in \{1, 2, \ldots Np\}\backslash\{i\}$) by using a without replacement roulette wheel strategy based on the probabilities of $R_p(i, :)$. The donor vector **v**' is then generated as follows:

$$\mathbf{v}' = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}).$$

A complete update of R and R_p is not required in every generation, as only the distances and probabilities for newly selected offspring are needed to be updated. The strategy experimentally showed to improve explorative mutation schemes such as DE/rand/1, but fails to show significant improvement when used over highly multimodal or hybrid functions or with greedy mutation schemes.

4.2.3 DE with Generalized Differentials (DEGD)

Ali (2011) proposed a new variant of DE, with a modified mutation scheme. The mutation scheme unlike DE/rand/1, does not use the scaled difference of vectors, rather it uses the difference of the scaled vectors, defined as follows.

$$\mathbf{v} = \mathbf{x}_{r_1} + F_1\mathbf{x}_{r_2} - F_2\mathbf{x}_{r_3}.$$

Note that when $F_1 = F_2 = F$, then the proposed equation matches the one of the DE/rand/1 (3a). The author also made a claim that instead of generating trial vectors for all of the population, it is better to do it for only the worse solutions. The author gave experimental results showing that as the best gets replaced in a selection less often than the worse solutions (say the m solutions from the bottom of the sorted list of population individuals, based on fitness and when the best is on the top), generating trial for it is not a very beneficial task, when high requirement of computational power is present. To enhance the chance of generating a better quality offspring, the trial vector generation process can be repeated at most say q times, for each target. If in the first $q - 1$ mutations, a successful trial is not generated, vector projection is used in place of mutation (two vectors will be

picked randomly from the population, and the worse will be projected on the better one) for the q^{th} time as follows:

$$\mathbf{v} = \left(\frac{\mathbf{x}_{r_1}^T . \mathbf{x}_{r_2}}{\mathbf{x}_{r_2}^T . \mathbf{x}_{r_2}} \right) \mathbf{x}_{r_2}.$$

4.2.4 DE with Ranking-Based Mutation Operators

Gong and Cai (2013) suggest that a mutation can be more fruitful if the base vector and one of the terminal vectors of the difference vector can be selected based on their fitness, while the third is selected at random. However, to make the process less greedy a vector will only have a probability to be selected as a base or a terminal. The authors proposed that the instead of randomized or proximity based approaches the probability for a vector to be selected in the mutation can be calculated from their fitness rank in the population. The proposed strategy is thus faster and less computationally expensive. The population is needed to be sorted based on their fitness values in ascending order (the best is at the top of the list). Then for the i^{th} solution, the rank value R_i is defined as $Np - i$. The probability for the selection of the i^{th} solution is then defined as $p_i = R_i/Np$. At the time of selecting three vectors for the DE/rand/1 strategy, the first two vectors are chosen proportionally to their probability of selection.

4.2.5 Intersect Mutation DE (IMDE)

To improve the search capability and convergence of regular DE, Zhou *et al*. (2013) proposed two new variations (the first focusing on exploration, while the second on exploitation) with modified mutation and crossover schemes. In each generation the population is first sorted. The best *M* vectors are kept in a set *B* and the remaining members are included in a set *W*, indicating better and worse solutions. In the first variation of DE, different mutation and crossover strategies are prescribed for the set *B* and *W*. For any individual in the set *B*, the mutation is done in a similar manner to rand/1; only the base vector is randomly selected from the set *W*, while the two vectors needed to calculate the difference are selected from the set *B*. The crossover is similar to the binomial crossover in nature; however, a donor can only take part in the crossover if it is fitter than the target, else the trial will be the same vector as the target. For a vector in the set *W*, the mutation is again performed similarly as rand/1, only the base is taken from set *B*, and the other two vectors are selected from *W*. Regular binomial crossover is applied in this scenario. In the second proposal only a new mutation is advised for the members of set *B*, while the rest of the algorithm is similar to the first variant. Here, the base and the left terminal of the difference vectors are selected from set *W*, while the third vector is picked from set *B*.

4.2.6 Neighborhood and Direction Information based DE (NDi-DE)

Understanding and utilizing neighborhood and directional information is important for a DE population to search efficiently. Following this line of thought, Cai and Wang (2013) suggested an improvement of the DE mutation strategy by introducing the guidance of neighborhood and directional information. They first defined a Neighborhood Guided Selection (NGS) scheme for selecting the base and difference vectors for mutation. For generating the donor of the i^{th} target, NGS first assigns a probability to each vector of the population. The probability for any vector with index j is calculated as:

$$p_j = 1 - \frac{d(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{j=1}^{Np} d(\mathbf{x}_i, \mathbf{x}_j)},$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ is the Euclidean distance between vectors $\mathbf{x}_i$, and $\mathbf{x}_j$. The algorithm uses a roulette wheel method to select three vectors in proportion to their probabilities from the population. The base vector is taken as the

winner of the tournament between the chosen vectors; the other two are subtracted to form the difference vector ($\mathbf{x}_{difference}$). NGS by these techniques exploit the neighborhood information and also selects the better vector as the base vector. The direction information is included by using a Direction Induced Mutation (DIM) strategy. DIM introduces a new vector known as the direction vector and adds it with the base vector and the scaled difference vector obtained through the neighborhood-guided selection process. For the mutation of the i^{th} target, DIM first identifies the nearest worst and nearest best solution of the target vector. The best near neighbor is defined to be the j^{th} vector, for which the ratio of the fitness difference between $\mathbf{x}_i$ and $\mathbf{x}_j$ and the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is the maximum. The nearest worst neighbor is defined to be the j^{th} vector, for which the ratio of the fitness difference between $\mathbf{x}_j$ and $\mathbf{x}_i$ and the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is the maximum. Let the best nearest neighbor be called as $\mathbf{x}_{i\_best}$ and the nearest worst to be $\mathbf{x}_{i\_worst}$. Three types of directional properties namely, Directional Attraction (*DA*), Directional Repulsion (*DR*) and Directional Convergence (*DC*) can be calculated from these vectors, and they are defined as follows:

$$\boldsymbol{DA}_i = I_{DA}\left(\mathbf{x}_{i_{best}} - \mathbf{x}_i\right),$$
$$\boldsymbol{DR}_i = -I_{DR}\left(\mathbf{x}_{i_{worst}} - \mathbf{x}_i\right),$$
$$\boldsymbol{DC}_i = I_{DC1}\left(\mathbf{x}_{i\_best} - \mathbf{x}_i\right) - I_{DC2}\left(\mathbf{x}_{i_{worst}} - \mathbf{x}_i\right).$$

The I_{DA}, I_{DR}, I_{DC1} and I_{DC2} are scaling factors which can be controlled to limit the influence of the directional vector. The rand/1 mutation scheme with DIM is as follows:

$$\mathbf{v} = \mathbf{x}_{base} + F\mathbf{x}_{difference} + \boldsymbol{DT}.$$

Here, ***DT*** can be ***DA***, ***DC*** or ***DR***. The algorithm is though simple and empirically proven to be effective. The choice of scaling parameter and directional information are problem and mutation strategy dependent respectively.

4.2.7 Multi-objective Sorting-based Mutation Operators for DE

Considering both the fitness and the population diversity to achieve a proper balance between exploration and exploitation, a bi-criteria mutation scheme is proposed by Wang *et al*. (2014). The algorithm uses two objective functions, one is the actual function to be minimized and the other is the summation of the pair-wise Euclidian distance between the individuals of the population. In each generation the solutions are subjected to a non-dominated sorting defined by Deb *et al*. (2002). This type of sorting generates a set of mutually non-dominated (in Pareto sense) solutions, called the Pareto optimal front. Thus, to successfully rank each individual of the population, another round of sorting is required within each front. This second sorting can be done based on a randomly picked objective function. The rank of the i^{th} individual is finally determined as $R_i = Np + 1 - i$, and an associated probability is calculated as $p_i = R_i/Np$. A simple roulette wheel model is proposed using the generated probabilities to select the parents for mutation.

4.3 DE with New or Improved Crossover Strategies

Wang *et al*. (2012) proposed the use of an orthogonal crossover operation instead of the popular binomial or exponential to enhance the searching capability of DE. The authors adapted the orthogonal crossover with a quantization technique (QOX) proposed by Leung and Wang (2001) who used it with GA. QOX can efficiently search the hyper rectangle formed by the donor and the target, using *M* function evaluations making it computationally expensive. To alleviate the situation, QOX is used only once in every generation, for a randomly selected target and its donor vector.

Binomial crossover though very effective and widely used in DE is not rotationally invariant. Recently, Guo and Yang (2015) suggested that a rotationally invariant crossover (similar in nature to the one proposed in (Wang *et al*, 2014)) will more successfully follow the function landscape and will generate better trial vectors. The authors pointed out that before crossover the co-ordinate system can be rotated to make the function landscape pseudo separable. The crossover can be performed on this system and then the system can be again rotated back to the original search space. Such a co-ordinate system can be reached by using the eigenvectors of the covariance matrix of the population as a rotation matrix. The target and the donor vectors are rotated by multiplying them with the matrix of the eigenvectors, and the crossover is performed on these rotated vectors. After the trials are generated they are multiplied by the conjugate transpose of the rotation matrix to bring them back in the original co-ordinate system. However, if all individuals undergo the rotation-invariant crossover, the DE population may not be able to reach the global basin of attraction quickly. Hence, in each generation, the individuals undergo crossover either in rotated or in original co-ordinate system with a pre-defined probability (which actually becomes a control parameter of the algorithm). Due to the need for computing the spectral properties of a covariance matrix, the algorithm may become computationally expensive for large scale optimization problems.

4.4 DE with Subpopulation and Cluster-based Improvisations

Some recent DE variants start with multiple sub-populations and some partition the initial population into a number of clusters which then act like sub-populations and take part in the evolution process. Below we present a few major single-objective DE algorithms that use multiple sub-populations and clustering techniques. As will be evident, such approaches are specially useful on multimodal landscapes with widely separated basins of attractions as different subpopulations can undertake detailed search within different basins simultaneously.

4.4.1 The DE/Cluster Algorithm

Li and Zhang (2011) proposed a subpopulation based DE, where the subpopulations are found by a clustering technique. The clustering technique (agglomerative hierarchical) is simple to implement, can create dynamic subpopulations, and is adaptive to the change of the population. However, the raw clusters may not be suitable for evolution (for example they may be too small). To circumvent this, the authors made some suggestions. All the clusters with a single element are combined together (called as *SPEX*). If the current best is a not member of this cluster, then it can be used to maintain the diversity of the population and the explorative capability of the algorithm. For each of the clusters, which have more than one member but do not have enough members to perform mutation, the algorithm maintains a pool of solutions. The pool is updated after every generation and helps a cluster by supplying required members to perform a mutation.

4.4.2 Clustering based DE (CDE)

To enhance the process of the generation of new population, Cai *et al*. (2011) proposed to use a one step *k*-means operation alongside DE trial vector generation strategy. The authors pointed that a single step of *k*-means can actually be viewed as a collection of multi parent crossover outputs. The one step *k*-means will start with *k* randomly selected vectors from the population (P), which will act as the initial cluster centers. The number of clusters *k* will be randomly selected as an integer from the range $[2, \sqrt{Np}]$. The cluster centers will then be calculated following a single iteration of the steps of the *k*-means algorithm. Let the new cluster centers form a set *A*, and also let another set of randomly chosen vectors from the population be *B*. The algorithm will select *k* best solutions from the set $A \cup B$ forming another set say *B'*. The new population will be $(P \backslash B) \cup B'$. As the clustering step is costly, it will only be applied after certain intervals. At the end of one such interval, after

updating the population with DE, the clustering will take place on the new population to further update it before being used in the next generation.

4.4.3 Clustering-based DE with 2 multi-parent crossovers (2-MPCs-CDE)

Liu *et al*. (2012) made further modification to the clustering-based DE proposed in (Cai *et al*., 2011), where they introduced two multi-parent crossovers over the one step *k*-means to generate trial vectors. They also introduced a small alteration in the DE/rand/1 scheme by imposing the condition that the base vector, selected from the current population, must be fitter than the target vector. From the beginning, the population is let evolve for every *m* consecutive iterations with the modified DE/rand/1/bin scheme. After every *m* iterations, the one step *k*-means and the new crossover schemes are applied on the corresponding population. The one-step *k*-means algorithm is used to generate a set C_{center} of the cluster centroids which are used in the second multi-parent crossover to create *k* individuals by Gaussian mutation. The value of *k* is chosen as $\sqrt{Np}$ by following other existing empirical studies. For each of the individuals (targets) of the current population (*P*) the first multi-parent crossover selects $k+n$ (*n* is taken as 2) random numbers (α) in the range [-1, 4], such that their summation becomes 1. For each of the *k* cluster centers ($\boldsymbol{C}_j$ where $j = 1, \dots, k$), a new vector ($\boldsymbol{C}_j$') is generated by adding with it, a scaled difference of two randomly picked vectors from the population. For *n* number of times, perform the prescribed rand/1 mutation on the population to generate *n* new vectors ($\mathbf{v}_j$). The new trial vectors are generated as $\mathbf{v}_i' = \sum_{j=1}^{k} \alpha_j \boldsymbol{C}_j' + \sum_{j=1}^{n} \alpha_{j+k} \mathbf{v}_j$. For each of the targets in the population a second multi parent crossover is performed by first selecting $k+m$ (*m* is taken as 1) random numbers (*b*) in the range [-0.1, 1.5], such that the summation becomes 1. Let $\boldsymbol{C}_{center}$ be the mean of the set *C*. The algorithm generates *k* new vectors as $\boldsymbol{C}_j' = \boldsymbol{C}_{center} * (1 + N(0,1))$. As in the first multi-parent crossover, here also other *m* vectors ($\mathbf{v}_j$) are generated by using the modified DE/rand/1 scheme. The new trial vector $\mathbf{v}_i^{"}$ is formed with the similar weighted sum technique, used in the first multi-parent crossover with the *b* values being used as weights. The population is updated by replacing the i^{th} target with the fitter vector between $\mathbf{v}_i'$ and $\mathbf{v}_i^{"}$. Let the results of the multi-parent crossovers form a new set *K*. The new population for the next generation is selected as the best *Np* individuals from the set $P \cup K \cup C$. The interval of generations after which these additional crossovers will be applied, is a crucial problem-dependent parameter of the proposed algorithm.

4.4.4 Multi-population DE with Balanced Ensembles

Ali *et al*. (2015) proposed multi-population DE with balanced search (mDE-bES) used to boost the diversity in the DE population to handle large-scale optimization problems. The population is divided into subgroups where each of these subgroups follows different mutation and update strategies. The authors introduced a new mutation strategy that support the search process by using information either from the best individual or from a random individual, probabilistically. It uses a linear combination of vectors to produce a base vector. All strategies select individuals using a strategy that rank individuals based on their fitness. Randomly selected individuals are used to migrate between the subgroups periodically to support the evolution process.

4.5 Population Topology and Population Diversity Guided DE

Population topologies can intensely influence the search by a metaheuristic continuous parameter optimizer as such topologies control the flow of information from an individual to its neighbors. This is quite evident from the research on various neighborhood topologies in another very popular algorithm well-known as Particle Swarm Optimization (PSO), (see for example Mendes (2004)). Similarly sensing the population diversity in course of the search can provide important guidelines to modify the search moves efficiently. This sub-section

reviews a few interesting and recent single-objective DE variants that involve population topology or diversity guided search mechanisms.

4.5.1 Improved DE with PBX-α Mutation Operator and Population Topologies

Dorronsoro and Bouvry (2011) presented a survey of such existing models and proposed five new variants of DE with distinct population models, suitable for various problems. They also suggested a new mutation operation inspired by the PBX-α operator used in GAs (Lozano *et al*., 2004). In the synchronous update model of DE, the selected solution vectors form an auxiliary population. Only after *Np* vectors are selected, the auxiliary population replaces the original. This is useful for exploration. However the new individuals have to wait for interacting with the population, and thus, the exploitation is reduced. In all of the following variants, a donor can only be generated by using neighboring solutions. In distributed DE, the entire population is partitioned into sub-populations, and an independent DE is run on each of them. After a generation, the information among the sub-populations is exchanged via a migration scheme. In cellular DE (Dorronsoro and Bouvry, 2011), the population is arranged in a mesh structure. Similar to the cellular DE, the hierarchical DE is also modeled to facilitate exploitation, the only difference is that the population structure is ranked, based on the fitness level (the best is on the top), and after each generation the solutions are reassigned a position guided by their current fitness values. The advantage is, the fitter individuals will always have a good surroundings of neighboring solutions and will have the opportunity to generate a better trial vector. The random topology DE will create a pseudo random graph with the solutions as its vertices. The neighborhood will be established by the connecting edges. The population topology will be fixed at the initialization stage and will only be rearranged if the population remains unchanged in any generation. In the final proposed variant called small world DE, the authors created a ring topology with each solution having *k* neighbors. After generating the graph for each vertex, an edge connecting with its neighbor will be replaced with an edge connecting to any random solution with a very low probability. The population models studied by the authors have been schematically shown in Figure 1. The proposed mutation is defined for the j^{th} dimension of the i^{th} target is defined as follows

$$I_j = x_{r_2,j} - x_{r_3,j},$$
$$up_{i,j} = \min\left(max_j, x_{r_1,j} - I_j\alpha\right),$$
$$low_{i,j} = \max\left(min_j, x_{r_1,j} + I^j\alpha\right),$$
$$v_{i,j} = x_{r_1,j} + G(0, 0.1).F.\left(up_{i,j} - low_{i,j}\right).$$

The value of α is adapted, by either picking it randomly from [0.2, 0.8] with a probability 0.1, or kept as the previous one.

4.5.2 Dual-population DE (DP-DE)

Zhong *et al*. (2013) proposed a dual-population DE (DP-DE) to control exploration and exploitation capabilities of DE. The two populations are used to serve different purposes in the search process. One population (*GP*) uses an explorative strategy and maintains diversity while the other (*LP*) performs an exploitative search over the neighborhoods. A new migration strategy was proposed after a regular selection operator to facilitate an interaction between the two populations. In DP-DE different mutation strategies are applied on the populations based on the purpose. For example, for *GP* population, *F* and *Cr* are chosen at random from the bounds (0.01, 1) and (0, 1) respectively, with DE/rand/1 or DE/current-to-*p*best/1 (Zhang and Sanderson, 2009b) mutation strategies and a binomial crossover operator. To further divert the population of *GP*, a mutation is performed which randomly changes a dimension of a vector with a certain probability say *pm*. For *LP* population, *F* is randomly picked from the standard normal distribution with $Cr = 0$ and DE/best/1

mutation is used to perform a local greedy search (In order to keep the search efficiency alive in the greedy scheme, the two random vectors needed in best/1 were picked from *GP* rather than *LP*). Following selection in both populations, the bidirectional migration is applied. First the best vector from *GP* ($\mathbf{GP}_{best}$) and two vectors (the best and the worst, called $\mathbf{LP}_{best}$ and $\mathbf{LP}_{worst}$, respectively) from *LP* are selected from the population and then the following operation is performed.

$$If\ f(\mathbf{GP}_{best}) < f(\mathbf{LP}_{best})\ then\ \mathbf{LP}_{best} = \mathbf{GP}_{best},$$
$$Else\ if\ f(\mathbf{GP}_{best}) < f(\mathbf{LP}_{worst})\ then\ \mathbf{LP}_{worst} = \mathbf{GP}_{best},$$
$$If\ f(\mathbf{GP}_{best}) > f(\mathbf{LP}_{best})\ then\ \mathbf{GP}_{best} = \mathbf{LP}_{best}.$$

The algorithm used two extra parameters, namely the p for the DE/current-to-$p$best/1 mutation strategy and pm for further diversifying *GP*. However, the algorithm is sensitive to the variation of these parameters and it is hard to establish any guideline about choosing their values except suggestions of an empirical range and avoidance of extreme values.

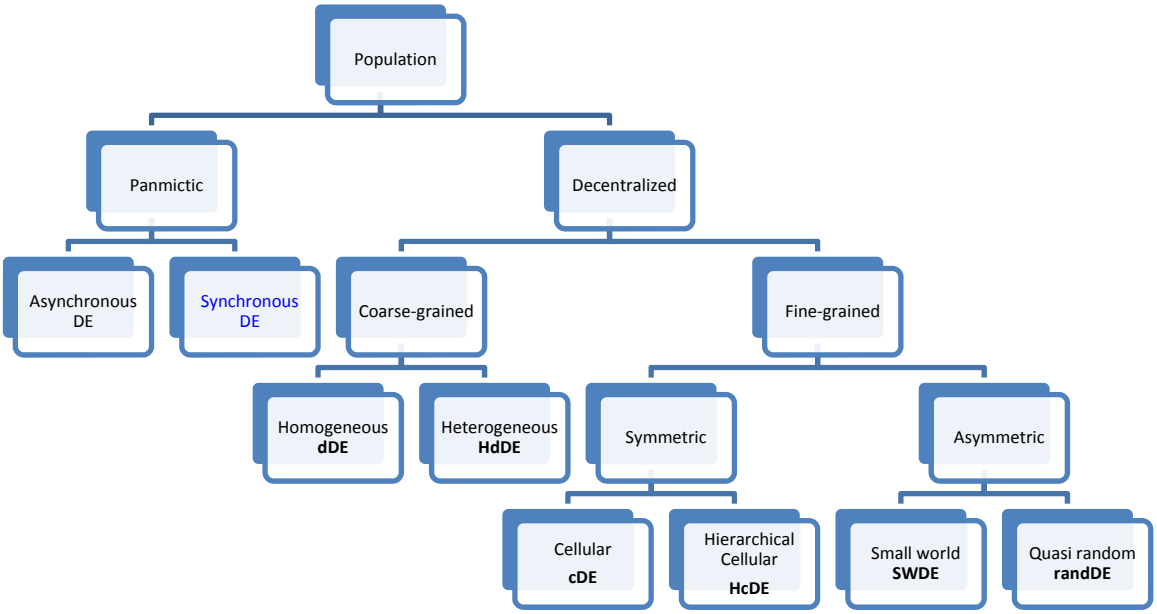

Fig. 1: Hierarchy of population models studied in (Dorronsoro and Bouvry, 2011)

4.5.3 DE with Auto-Enhanced Population Diversity

In DE, the population diversity needs to be maintained to stop pre-mature convergence, alongside avoiding any stagnation to keep the algorithm in a fruitful searching mode. Yang *et al.* (2015) designed an automatic population enhancement scheme that will check each dimension to identify a convergence and will diversify that dimension to a satisfactory level; this will also aid DE to escape from a local minimum and stagnation. To quantify diversity, mean and standard deviation for each of the dimensions is calculated for the population. A lower standard deviation in a direction indicates lower diversity, so for each dimension, a threshold is maintained, if the standard deviation is found to be below the threshold, the dimension is called converged. The threshold for the j^{th} dimension is $w_j = \min(T, \theta_j)$. Here $T = 10^{-3}$ and if mean and standard deviation for the j^{th} dimension is m_j and sd_j then

$$\theta_j = \begin{cases} |m_j - MR_j| * T\ if sd_j \leq T, \\ T\ otherwise. \end{cases}$$

Here MR_j is the m_j of the time when the last diversity operation in j^{th} dimension was performed, initially set to the mean of the initial population. If the dimension converges towards the same point as previous then $|m_j -$

$MR_j| * T$ will be chosen as the threshold, further decreasing it, pointing to the original convergence. On the other hand if the dimension is converged towards a new point the threshold will be set to *T*. To identify stagnation a counter for each dimension can be maintained which will increase by 1 after a generation if the mean and standard deviation doesn't change in the generation, else reset to 0. If this counter reaches a threshold *UN* then the dimension can be called stagnant. But diversifying a dimension may not lead to convergence, and for non-separable problems diversifying a single dimension may lose the progress of the entire run. To solve this issue, an indicator of diversification is only set to one, when all the dimensions are converged or stagnant. But, for a high dimensional problem, this is hard to achieve, thus the population will also be diversified with a very small probability (taken as 10^{-3}) at a generation. A dimension is diversified by reinitializing it in each solution with a value in the allowable range [max_j, min_j]. This is done by generating a value for the i^{th} vector in the j^{th} dimension as follows $x_{i,j} = min_j + randn(\mu_j, \sigma_j) * (max_j - min_j)$. The function *randn* randomly picks a number in the range [0, 1] from a Gaussian distribution with a mean μ_j and standard deviation σ_j , which are defined as:

$$\mu_j = \frac{m_j - min_j}{max_j - min_j},$$

$$\sigma_j = \exp\left(\frac{-aK}{d}\right) * \overline{\sigma_j},$$

$$\overline{\sigma_j} = \max(\mu_j, 1 - \mu_j).$$

The value of *K* is the number of current function evaluations and a is empirically found to be 0.0005. The value of *K* will ensure that with the progress of the DE generation the diversification will happen more close to the previous generation, believing that the DE is actually converging with increasing generations.

4.6 DE with Representation of Individuals in Probabilistic Space

4.6.1 Compact DE

The difficulty of using evolutionary algorithms in intelligent devices used in regular life arises from the memory requirement for storing the population and the computation power to process it. To solve this issue, a class of EAs known as compact EAs was developed with a reduced representation of the population by its statistical properties rather than as a collection of individuals. Mininno *et al*. (2011) developed a compact variant of the DE algorithm, called cDE, by adopting the general operations and workflow of the common DE. cDE at the time of optimizing a *d*-dimensional problem, represents a population by a multi-dimensional, truncated and normalized Gaussian distribution, storing only the mean and variances of each dimensions. cDE first initialize the means to be zero and the variances to be high, to mimic the behavior of a uniform distribution and pseudo-randomly samples an initial "elite" solution from it. In an iteration of the algorithm, cDE samples a number of solution vectors from the distribution as required by the mutation strategy (for example three vectors for DE/rand/1 mutation scheme) and generates the donor vector. Crossover is performed between the elite and the donor to produce a trial vector. Selection is done between trial and elite based on their fitness values and the selected vector replaces the elite. The newly selected elite and the loser from the selection process both will be used to update the mean and variances, for the next iteration. The main advantages of cDE are firstly the low space requirement of at most 6*d* units (2*d* for the distribution, 3*d* for mutation, and *d* for elite), and secondly the computationally expensive mutation, crossover and fitness calculations are only done once to generate or select a single vector per iteration. However, the dimensions are assumed to be independent, which simplifies the problem by representing the covariance matrix as diagonal. In reality, this may become a very hard constraint

and the algorithm may not completely identify the global peaks in a multimodal functional landscape with strong linkages among groups of variables.

4.6.2 Stochastic Coding DE (SDE)

Zhong and Zhang (2012) proposed to use a multivariate Gaussian distribution for each individual, instead of using the individual solutions themselves. This coding technique is claimed to be robust and can perform a region-wise search with ease. Each individual of the population in the new coding scheme is represented by a mean vector of length *D* and a square covariance matrix. The mean vector (μ) is initialized with a random value in the range of the variables, and the covariance matrix (Σ) is initialized as a diagonal matrix. In each generation an update operator is applied on the population. For this, first a temporary solution is sampled by $\mathbf{u}_i = \boldsymbol{\mu}_i + S\boldsymbol{Z}$. Here $\Sigma = SS^T$, S is upper triangular, and $\boldsymbol{Z}$ is a vector sampled from a standard normal distribution. Algorithm's performance can be further improved by moving the mean towards the best so far mean, and enhancing the diversity. After sampling the solution $\boldsymbol{u}_i$, a corresponding neighborhood is generated, where each neighbor $\boldsymbol{n}_i$ is produced by replacing a single dimension (randomly picked) of the mean by the dimension of $\boldsymbol{u}_i$. After generating all the neighbors, the ones which are better than the given individuals are selected and kept in a set say *B*. If *B* is non-empty then the covariance matrix of *B* replaces the covariance matrix of the individual, and the mean is replaced by the mean of the best neighbor. After this a regular DE/rand/1/bin is used to update the mean vectors, and later the covariance matrix. A major challenge of this algorithm is the high computational cost, even after the authors' proposal of making only the top *M* individuals of the population, participating in this update.

4.6.3 Gaussian Bare-bones DE

To reduce the effects of control parameters and to reduce the importance of selection or adaptation, Wang *et al*. (2013) proposed a two-fold variation of DE, influenced by the concept of the bare-bones particle swarm algorithm (Kennedy, 2003), and named them as the Gaussian Bare-bones DE (GBDE) and Modified GBDE (MGBDE). In GBDE, the mutation and the selection of *Cr* have been updated, whereas in MGBDE the mutation step is further modified to include the advantage of the DE/best/1 strategy. In GBDE, instead of regular mutation schemes, the donors are generated by picking them randomly from a Gaussian distribution with mean $\boldsymbol{\mu}$ and standard deviation σ. As the $\boldsymbol{\mu}$ and σ are calculated from the current population itself, the need for *F* is eliminated. For generating the donor of the i^{th} vector, $\boldsymbol{\mu}$ is calculated as the average of the best and the target vector, while σ is the modulus of the difference between them. The proposed mutation is explorative in the start, but as the generation increases the difference between the best and any individual will decrease and the average will go toward the best, encouraging exploitation. Regular binomial crossover technique will be used, but the value of *Cr* will be initially chosen from a Gaussian distribution of mean 0.5 and standard deviation 0.1. In later generations, *Cr* value will be individual dependent and will keep its current value for an individual if that solution has been updated in the current generation, else it will be reinitialized. In MBGDE the mutation strategy is taken to be either DE/best/1 or the Gaussian with equal probability. According to experimental results MBGDE outperformed GBDE. Due to the explorative nature of the mutation both of them succeeded in several multimodal functions, while failing to outperform other DE variants in some of the simple multimodal or unimodal cases.

4.7 DE with Parent Selection Framework

Classical DE does not have a parent selection criterion for mutation or crossover, as every individual has to undergo these two steps. However, the need for parent selection in DE has been advocated in works like (Sutton *et al*., 2007; Islam *et al*., 2012). Recently Guo *et al*. (2015) have proposed an interesting parent selection framework for DE. Guo *et al*. (2015) viewed stagnation as a situation in an iteration, where the algorithm neither converges to a fixed point, nor does it find any better solution. They tried to quantify stagnation as well. By simply monitoring the sum of distances of the individual vectors from the population mean, one can identify if the algorithm is converging to a fixed point. However, for high dimensional and complex multimodal functions it is very hard for DE to achieve a converged population. Nevertheless, it is possible to know if DE has failed to generate any better solution for a number of successive iterations. If that is the case, the population can be improved by adding offspring from the previously successful parents. For each individual of the population a counter is maintained which increases by one if it is not replaced in selection, else reset to 0, for being used in the next iteration. At the time of rand/1 mutation, if the counter of a target vector crosses a predefined threshold, three vectors are randomly picked from the archive of size *Np* containing the recently updated solutions; else the vectors are randomly picked from the current population. The archive is updated by adding the newly selected solutions replacing the oldest ones, after each generation.

4.8 Others

Apart from the most prominent DE variants briefly reviewed in this section under seven heads, there are several other modified DE schemes that have been and are being published on a regular basis since 2010. Some of these schemes that deserve mention are in order. Lu *et al*. (2014) modified the trial vector generation strategy proposed in CoDE (Wang *et al*., 2011b) and introduced a self-adaptive DE framework and called their algorithm DE with Surrogate-assisted Self Adaptation (DESSA). The proposed algorithm maintains a database or archive, which is built in the first few generations. The initial population is created randomly and until the database building is completed the evolution happens following the base algorithm and the generated populations are added in the archive. After the database is completely build, for each of the target vector, the trial is generated from a surrogate model (depending on the base algorithm used) constructed from the database. After producing the new population following regular selection, it is copied into the archive, replacing the old records of the database. Although DESSA uses the trial vector generation scheme of CoDE, any classical DE or variant of DE can also be used in its framework.

In 2011, Asafuddoula *et al.* (2011) proposed a variant of DE by introducing a new crossover technique (the center-based differential exponential crossover), a parameter adaptation scheme and an embedding of gradient based local search in the regular DE framework. This variant participated in the CEC 2011 competition on testing EAs with real world optimization problems and secured 4th rank. Brest *et al*. (2013) further improved the popular jDE algorithm (Brest *et al*., 2006) by introducing a subpopulation dependent trial vector generation strategy, a mechanism for preserving the diversity, and an aging scheme to detect possible stagnation in the original jDE framework. Zhabitskaya and Zhabitsky (2013a) proposed a variant of asynchronous DE with a new crossover scheme based on an adaptive correlation matrix computed on the members of the current generation and indicating the groups of correlated variables meaningfully. Recall that in asynchronous DE, only a single target takes part at a time, in the trial vector generation scheme, and the generated trial vector if selected, replaces the target, and immediately becomes a member of the population. The authors used the framework of asynchronous DE with independent restart algorithm (Zhabitskaya and Zhabitsky, 2013b), with a JADE type

parameter adaptation, to apply the proposed crossover strategy. Poláková *et al*. (2014) tried to address the issue of stagnation, and proposed a way to identify such a situation. They used the same competitive DE framework of Bujok *et al*. (2014). However, only the twelve strategies from the work of Tvrdík *et al*. (2012) are used to build the strategy pool. Yu *et al*. (2014b) proposed a variant of DE, where the greediness of the mutation strategy can be adapted with the progress of the algorithm. The authors modified the DE/best/1, DE/current-to-best/2 and DE/best/1 mutation strategies, by replacing the best vectors in equations (3b), (3c) and 3(d), respectively, with a randomly selected vector from the top k solutions. Clearly the greediness of the mutation can be controlled by tuning k, where a larger k will lead the proposed mutation towards the random mutation strategies, and a smaller k will result in the greedy ones. Xu *et al*. (2014) proposed a DE variant, with a replacement policy for stagnation. The replacement can be done in either for an individual, or for the population itself.

Considering the fitness at the time of mutation is good for creating a better offspring, however it also contains the risk of uncontrolled exploitation and pre-mature convergence. To aid this issue, Li et al. (2014) proposed a new mutation scheme, such that a balance between exploration and exploitation can be maintained. In each generation, the population is divided into three classes based on the fitness values. The individuals are re-indexed, such that the lower index values have better fitness and belong to the first class. The second class contains the medium performers, and the worse individuals having the higher indices belong to the third class.

Mohamed (2015) presented a new triangular mutation rule for DE by using a convex combination of the triplet defined by the three randomly chosen individuals and the difference vector between the best and the worst individuals among them.

5. DE in Complex Optimization Scenarios

For over the last five years, there has been a significant advance in research to adopt DE for optimization in complex environments that include optimization with nonlinear constraints (equality and inequality), multiple objectives, dynamic and noisy fitness landscapes, multi-modality and very high dimensionality of the search space. This section provides an overview of the DE approaches developed over the past five years to tackle such optimization processes.

5.1 Constrained Optimization

Real world problems often deal with the optimization of an objective function, subject to additional inequality and equality constraints. These are known as *constrained optimization* problems. A typical problem is concerned with finding a feasible d dimensional vector $\mathbf{x} = [x_1, x_2, \dots, x_d]$ in the search space $\mathcal{S}$, defined by the variable-wise boundary conditions $x_{max,j} \leq x_j \leq x_{min,j}$ $(j = 1, \dots, d)$, where $x_{max,j}$ and $x_{min,j}$ are the upper and lower bounds of x_j, respectively. The feasible region $\mathcal{F}(\subseteq \mathcal{S})$ is defined by a set of m additional *inequality constraints* $g_j(\mathbf{x}) \leq 0$ $(j = 1, \dots, q)$ and *equality constraints* $h_j(\mathbf{x}) = 0$ $(j = q+1, \dots, m)$.

A DE algorithm with an Ensemble of Constraint Handling Techniques (ECHT) was proposed by Mallipeddi and Suganthan (2010), where each constraint handling method has its own population. A distinguishing feature of the ECHT is the utilization of every function call by each population associated with each constraint handling technique. In constraint problems, depending on several factors such as the ratio between feasible search space and the whole search space, multimodality of the problem, the chosen EA, and global exploration/local exploitation stages of the search process, different constraint handling methods can be effective during different stages of the search process. ECHT is able to adapt to the requirements over evolution.

Mezura-Montes *et al*. (2010) presented an empirical study on the performance of DE in constrained optimization problems. They performed an experiment with four DE variants (DE/rand/1/bin, DE/best/1/bin, DE/target-to-rand/1 and DE/target-to-best/1) and tested their performance in different dimensional (high and low) and structured problems (with or without equality constraints). Based on the experimental results, they proposed a combined search algorithm where in the first stage a DE/rand/1/bin will run to find a feasible region, after that finer search towards optimum will be performed by DE/best/1/bin strategy. Liu *et al*. (2010) developed a hybrid of DE with PSO to solve constrained numerical optimization problems based on penalty functions and feasibility rule (Powell and Skolnick, 1993; Deb 2000). A single-objective DE framework with a fuzzy control system influenced quantification of feasibility of a solution was proposed in (Wang and Li, 2011). In this scheme, each of the solutions is given a satisfaction level based on the constraint violation. The authors kept the basic DE framework with rand/1/bin and only modified the selection procedure by comparing the trial vector with the target vector using the α-comparison operator. The value of α (a scalar parameter) is made to be iteration dependent such that in later stage the increment in number of feasible solutions is encouraged. Sarder *et al*. (2011) proposed a constrained optimizer based on DE by incorporating the idea of gradient-based repair with a DE/rand/1/bin scheme. If the individual candidate solutions generated by DE are infeasible the algorithm calls for a gradient-based repair to convert those into feasible solutions. Thus, as the generation count increases, the ratio between feasible search space and the whole search space is increased.

Mohamed and Sabry (2012) presented a modified DE to handle constrained optimization problems. This variant of DE comes with a mutation scheme, a strategy for choosing the parameters and a constraint handling policy. A new type of mutation is proposed where the base vector is added with the scaled difference of the global best and the worst vector. The selection process is modified to select a trial based on any of the following three criteria. The trial vector is accepted for next generation (1) if it is fitter than target (when both are feasible), (2) if it has lesser penalty for constraint violation than target (when both are infeasible), or (3) if it is feasible while the target is infeasible. The problem of constrained optimization demands not only to optimize a function but also to respect the constraints imposed upon its dimensions. A way to tackle this kind of problems is to quantify the overall constraint violation of a solution and try to minimize it alongside optimizing the function. Wang and Cai (2012) proposed such a bi-objective framework for constrained optimization with DE. The penalty function used to measure the constraint violation of a solution, is similar to the one used in (Liu *et al*., 2010). The authors modified the DE framework to satisfy their requirement.

Wang and Cai (2011a) integrated a $(\lambda + \mu)$ DE with an improved adaptive trade-off model to handle constrained optimization problems efficiently. In $(\lambda + \mu)$ DE, each individual is made to produce three offspring by using 3 mutation strategies (the rand/1 strategy, the current-to-best/1 strategy, and the rand/2 strategy) and after the parent and offspring populations are combined, the selection method of $(\lambda + \mu)$ ES (Evolution Strategy) is used to select the survivors for the next generation. Jia *et al*. (2013) continued the work further by proposing some modifications to the basic algorithm. In their proposal, the dimensional constraint violation is calculated in a similar manner as in (Mohamed and Sabry, 2012). Two approaches are used to measure the penalty of a vector, depending on the difference among the maximum and the minimum values of the constraint violations over all the dimensions. If the difference is significant, then normalized mean as per (Mohamed and Sabry, 2012) is taken, else simple summation as (Liu *et al*., 2010) is used to find the penalty.

Believing a fitter individual to be more preferable for generating the trial and for convergence of the algorithm, Gong *et al*. (2015) used a rank-based approach to select the vectors participating in a mutation to generate a trial vector. But, for a constrained optimization problem, considering only the fitness value is not

enough, a quantization of the infeasibility of a solution is also needed. The authors achieved such a quantization by a penalty function based on mean constraint violation. Gao *et al*. (2015) formulated a constrained optimization problem as a bi-objective optimization problem (by treating the degree of constraint violation as an additional objective) and used a Dual Population DE (DPDE) to solve the same. DPDE maintains two populations, one containing the infeasible solutions while the other holds the feasible ones. At the time of mutation following the rand/1 strategy, two vectors (the base and a terminal of the difference) are selected at random from the same subpopulation of the target, and the third vector is selected at random from the entire population. This modified mutation allows information sharing through the difference vector, between the two populations. However, the only problem with this type of approach is the mutation requires a minimum of three vectors in any subpopulations. The authors suggested if any subpopulation has the cardinality less than three, then it is no longer used and a simple DE is executed on the other, to either minimize the penalty or maximize the fitness. Recently Saha *et al*. (2015) proposed a fuzzy rule based constraint handling technique by using DE as the base optimizer.

Wu *et al*. (2015b) proposed an equality constraint and variable reduction strategy (ECVRS), which exploits the equations expressing equality constraints to eliminate equality constraints as well as variables of constrained optimization problems. It was shown that ECVRS can significantly improve the efficiency of DE when solving constrained optimization problems with equality constraints.

5.2 Multi and Many-objective Optimization

Multi-objective optimization involves finding the best trade-off among more than one objective. For a nontrivial Multi-objective Optimization Problem (MOP), there is no single solution that simultaneously optimizes each objective. In that case, the objective functions are said to be conflicting, and there exists a (possibly infinite) number of Pareto optimal solutions. A solution is called non-dominated, Pareto optimal, or Pareto efficient if none of the objective functions can be improved in value without degrading some of the other objective values. Many-objective optimization can be loosely seen as a large-scale version of MOPs involving 4 or more objective functions. Standard Evolutionary Multi-objective Optimizers (EMOs) face several problems when applied to many-objective problems including large non-dominated fraction of the population, computationally expensive evaluation of diversity measures, inefficient recombination operators, difficulty in representation of the trade-off surface and so on. Over the past 5 years, EMO researchers have also paid attention to DE. Prior to 2010, there have been no significant DE-based approaches towards many-objective optimization involving more than 3 objectives. A few significant approaches to modify and apply DE for multi and many-objective optimization are in order.

Zhong and Zhang (2011) proposed an Adaptive Multi objective DE with stochastic coding strategy (AS-MODE) where each individual in the DE population is represented, not by the exact solution, but by a multivariate Gaussian with a diagonal covariance matrix. A simple DE/rand/1/bin strategy is used for generating trial vectors. However, the vectors participating in the mutation process were chosen by using a tournament selection instead of picking them at random. The selection process involves a non-dominated sorting followed by the crowding distance based operation to rank the solutions of the set, from where top *Np* solutions are picked for the next generation. The algorithm, however, introduces 6 new parameters apart from the 3 usual parameters (*F*, *Cr*, and *Np*) of DE. Ali *et al*. (2012) proposed a new variant of DE, suitable for multi objective optimization. They used a different population initialization technique, by first generating two populations, each of size *Np*, and selecting the best *Np* from the combined set. The first population is generated by solutions picked randomly from the entire search space. The second population is formed by the opposite of the members

of the first one, in a manner similar to the opposition DE algorithm (Rahnamayan *et al*., 2008). The two populations are then combined and *Np* top solutions are selected by non-dominated and crowding distance based ranking (Deb *et al*., 2001, 2002). The mutation is performed using DE/rand/1 strategy. However, the non-dominated solution among the three vectors participating in mutation is selected as the base. An alternative approach to tackle the multi-objective optimization is to decompose it into several single-objective problems (Zhang and Li, 2007) by a linear or non linear weighted aggregation of the multiple objectives known as Multi-Objective Evolutionary Algorithms by Decomposition (MOEA/D). Zhao *et al*. (2012) introduced an ensemble approach to replace the tuning of neighborhood size parameter in the MOEA/D-DE (Zhang *et al*., 2009) and demonstrated that an ensemble of neighborhood parameters yielded an overall improved performance. The algorithm proposed by Venske *et al*. (2014) also follows the MOEA/D approach and uses an adaptive strategy selection. For generating the trial vectors, three strategies are used, two of them are the regular DE/rand/1/bin and, DE/rand/2/bin. The third one is DE/nonlinear (Sindhya *et al*. 2011), which has an advantage of not using any control parameter.

Qu and Suganthan (2011) improved the selection method and integrated with summation of normalized objectives based multi-objective differential evolution to solve multi-objective optimization problems. In the proposed selection method, a pre-selection process is added to remove the bad solutions and improve the convergence. The pre-selection is realized through using a reference point. All the solutions dominated by this reference point will be removed. The reference point is starting at the centre of the objective space and gradually moves to the origin along the search process.

Rakshit *et al*. (2014) developed a modified version of a popular multi-objective DE algorithm known as DEMO (Robič and Filipič, 2005), which can address MOPs in a noisy environment. The major problem of such type of environment is that the fitness value of an individual changes over sampling. To tackle this issue the authors proposed a simple alteration of the initialization and selection step of DEMO to apply three strategies. Firstly an adaptive sample size is suggested to measure the fitness of any individual in a noisy environment, secondly the significance of using of expected value and variance of fitness rather than simple averaging is established, and thirdly the authors prescribed a comparison technique that will deeply investigate the chance of a slightly worse trial to be placed in the Pareto optimal front.

Denysiuk *et al*. (2013) proposed a DE-variant for solving many-objective optimization problems called MyO-DEMR (Many-Objective DE with Mutation Restriction). The algorithm uses the concept of Pareto dominance coupled with the inverted generational distance metric to select the population for the next generation from a combination of the parent and offspring populations. The algorithm also utilizes a strategy for the restriction of the difference vector in DE mutation for improving the convergence characteristics over multi-modal fitness landscapes. Recently, Bandyopadhyay and Mukherjee (2015) proposed a many-objective optimization algorithm which periodically reorders the objectives based on their conflict status and selects a subset of conflicting objectives for further processing. The authors employed DEMO as the underlying metaheuristic evolutionary algorithm, and implemented the technique of selecting a subset of conflicting objectives using a correlation based ordering of objectives. The resultant method is called α-DEMO, where α is a parameter determining the number of conflicting objectives to be selected. DE has also been used as a base optimizer in the recently developed decomposition based MOEAs for many-objective optimization like (Jiang and Yang, 2015).

5.3 Optimization in Dynamic and Uncertain Environments

Several optimization problems in the real world are dynamic in nature. For these Dynamic Optimization Problems (DOPs), the functional landscape changes with time, i.e., optima of the problem change their locations

over time and, thus, the optimizer should be able to track the optima continually by responding to the dynamic environment (Nguyen *et al*., 2012). Practical examples of such situations are price fluctuations, financial variations, stochastic arrival of new tasks in a scheduling problem, machine breakdown, or maintenance.

DE has remained a popular choice for the DOP researchers since the development of Dynamic DE (DynDE) by Mendes and Mohais in 2005. Plessis and Engelbrecht (2012) improved the dynamic DE (DynDE) (Mendes and Mohais, 2005), by providing a method for faster convergence, and fixing the issue of diversity maintenance. The algorithm maintains multiple populations. At any moment only the best performer of them is evolved in its own, until a new population becomes the best with a better performance score. The continuous competition among the populations helps to improve the solution quality faster and provides quick convergence towards global optimum. Halder *et al*. (2013) proposed a cluster-based DE algorithm with external archive to address DOPs. The authors suggested clustering the entire population into subpopulations and evolving them differently, until a global solution is reached. The number of clusters (k) is an algorithmic parameter, and the authors proposed an adaptive technique by modifying it dynamically based on the algorithms performance. To tackle the dynamic environment, after every generation the best individual from each cluster is kept in the external archive. If at the end of a generation the fitness of the archived individual and the current best of the clusters mismatches, then an environmental change is inferred.

Kundu *et al*. (2013) proposed an improved variant of the Crowding DE (CDE) (Thomsen, 2004), for tracking multiple optima in a dynamic environment. The authors proposed to limit the participants of the mutation among the neighborhood of the target following an earlier work by Qu *et al.* (2012) on niching (to be discussed in Section 5.4). However, to reduce the rigidity of the approach they decided to select k vectors, using a roulette wheel model where the probability of selecting a vector is inversely proportional to its distance, from the target. To perform a simple DE/rand/1 mutation, 3 vectors will then be selected randomly from the k vectors, picked in the earlier stage. To detect a change in the dynamic environment, the authors suggested the use of a test solution. The test solution is not part of the population, so it does not participate in the algorithmic computations However, after each generation, its fitness is calculated. If there is any discrepancy between the fitness values of the test solution in two successive generations, then a dynamic change can be inferred.

Mukherjee *et al*. (2014) proposed a new dynamic DE algorithm, using clustering to generate sub-population, crowding based technique to maintain the diversity and local information, and a new crowding based archive to help the algorithm to adapt with a dynamically changing environment. Das *et al*. (2014) suggested a dynamic DE algorithm where they used the popular multi-population approach accompanied with two special types of individuals in each subpopulation to maintain the diversity. These individuals are known as Quantum or Brownian individuals and do not follow the DE rules. The algorithm also employs a neighborhood-driven double mutation strategy to control the perturbation and thereby prevents the population from converging too quickly. In addition, an exclusion rule is used to spread the subpopulations over a larger portion of the search space as this enhances the optima tracking ability of the algorithm. Furthermore, an aging mechanism is incorporated to prevent the algorithm from stagnating at any local optimum. Hui *et al*. (2014) proposed a niching based self-adaptive ensemble DE for solving Dynamic Optimization Problems. This method also integrated a modified multi-trajectory search operation to perform local search within selected niches.

MOPs can be dynamic as well involving more than one conflicting objectives that change continuously with time. Wan and Wang (2013) proposed a DE scheme equipped with an adaptive immigration scheme (to improve diversity) for tackling dynamic MOPs. DE has also been used as the base optimizer in a scalarization based dynamic MOEA that relies on controlled extrapolation and Pareto Front based nearest distance approach

(Biswas *et al*., 2014a). Dynamic Constrained Optimization Problems (DCOPs) constitute a unique class of optimization problems where the objective function as well as the constraint functions changes with respect to time. A first approach to solve DCOPs by using DE was taken by Pal *et al*. (2013). The authors integrated an improved offspring repair method with DE to maintain the diversity of the population. Ameca-Alducin *et al*. (2014) proposed a DE-variant by combining DE/rand/1/bin and DE/best/1/bin schemes and by using a memory of best solutions found during the search. Moreover, their algorithm allows addition of random immigrants to the population at each generation and use of a simple hill-climber-based local search operator to promote a faster convergence to the new feasible global optimum. Eita and Shoukry (2014) proposed a constrained dynamic DE (CDDE) algorithm where *Cr* and *F* are sampled from the range [0.5, 1] uniform at random to enhance the degree of exploration. Also, the authors suggested a novel hybrid simple constraint handling technique which combines two well-known techniques: feasible rules and adaptive penalty function.

5.4 Multimodal Optimization and Niching

Multimodal optimization techniques try to detect multiple global and local optima (as opposed to a single solution) of a function, so that the user can have a better knowledge about different optimal solutions in the search space and as and when needed, the current solution may be switched to another suitable one while still maintaining the optimal system performance**.** Niching is a generic term used mostly in connection with the multimodal optimization problems to denote methods for finding and preserving multiple favorable regions (niches) of the solution space possibly around multiple optima. DE has remained a very popular choice for multimodal evolutionary optimization since Thomsen's work on Crowding DE (Thomsen, 2004). As will be evident from the following discussions, unlike the previous decade, for last five years, there has been a surge of niching algorithms based on DE that rely on Euclidean neighborhoods and that have exhibited promising performance on modern multi-solution optimization benchmarks like those proposed in Qu *et al.* (2015).

To the best of our knowledge, Suganthan (1999) first introduced the concept of Euclidean distance-based geographical neighborhoods for the PSO algorithms to generate offspring. Following the same concept, Qu[1] *et al*. (2012) proposed a very simple yet efficient Euclidean distance neighborhood based mutation scheme and incorporated it within three multi-solution search approaches capable of performing niching by crowding (Thomsen, 2004), speciation (Li, 2005) and modified fitness sharing (Goldberg and Richardson, 1987) to form 3 new neighborhood mutation niching DE algorithms. The use of a global mutation operator (like DE/rand/1) is not suitable in case of multimodal optimization problems which require parallel and distributed convergence to distinct optima positions. To solve such problems, a number of localized searches in close niches are needed to be performed so that target vector, r_1, r_2, etc. and trial vector are all from the same neighborhood. Keeping the respective framework alive for these DE variants, the proposed mutation scheme selects three random vectors only from the parent's neighborhood and generates a donor by adding the scaled distance of two vectors with the third one. Strength of the proposal lies in its computational simplicity and not using any sensitive niching parameter except for the neighborhood size specified by the number of Euclidean distance neighbors (6 in this case) of the target vector to generate its trial vector. Even though neighborhood operations have been used for selecting next generation parents, Qu *et al*. (2012) seem to be the first to use Euclidean neighborhood for offspring generation in niching algorithms. For solving multimodal optimization problems, Epitropakis *et al*. (2011b) modified two popular mutation strategies of DE (rand/1 and rand/2) such that the base vectors to be perturbed in these schemes may be the nearest spatial neighbors of the current target vector $\mathbf{x}_{i,G}$. The resulting

[1] First submitted to *IEEE Transactions on Evolutionary Computation* in June, 2010.

mutation schemes (DE/nrand/1 and DE/nrand/2) exhibited niche formation without the need of additional parameters.

Besides the common niching techniques, another way to solve a multimodal optimization problem is to somehow formulate it as an MOP and focus on the solution set provided by the algorithm to identify multiple global and local optima. Basak *et al*. (2013b) proposed such a technique where they redefined the problem as a bi-objective one, and then used non-dominated sorting and dominated hyper-volume based sorting to filter the multiple solutions. The first objective function used in this technique is the actual objective function, while the second one is basically the mean distance among the solutions and is used to maintain diversity to ensure that multiple solutions from the entire search space are discovered. Liang *et al*. (2014a) proposed a variant of DE, where the population is driven to form close niches around the optima. Such techniques can only be successful when a solution vector will be evolved by using information from its neighborhood, or with individuals sharing the same local landscape. But also to maintain and improve the quality of the solution, the target needs to select better vectors to produce a good trial. To balance these two requirements, a modified Fitness and Euclidian distance Ratio (FER) is utilized in the algorithm. Epitropakis *et al*. (2013) proposed a niching DE algorithm by using a dynamic external archive to store the potentially good solutions discovered by the population so far. The algorithm used a control parameter adaptation scheme similar to JADE (Zhang and Sanderson, 2009a) and a re-initialization scheme to explore the search volume efficiently.

To address the multimodal optimization problems, Gao *et al*. (2014) proposed a cluster-based DE where the whole population is partitioned into subpopulations so that different subpopulations can locate different optima. Furthermore, the self-adaptive parameter control is employed to enhance the search ability of DE. The authors integrated the proposed multi-population strategy and the self-adaptive parameter control technique with two versions of DE: crowding DE and species-based DE. Biswas *et al*. (2014b) proposed a niching parameter free variant of DE for the multimodal optimization problem. The proposed variant employs a parent centric normalized neighborhood mutation and a synchronous population update mechanism. DE has also been used as the base optimizer for a very recently published niching method (Biswas *et al*., 2015) which is free of the explicit niching parameters like user-defined radius that requires information about the location and spacing of the peaks on a fitness landscape. This niching framework utilizes the concept of local information (distance and estimated gradient) from adjoining members to guide the search process. The authors improved the 3 niching DE variants proposed by Qu *et al*. (2012) by using their framework and reported promising results on the standard benchmark functions. Hui and Suganthan (2015) markedly improved the speciation based niching DE algorithm by integrating the arithmetic crossover operation. The resulting algorithm, called by the authors as Ensemble and Arithmetic Recombination-based Speciation DE (EARSDE) also could achieve a better exploitation of the individual peaks by applying neighborhood mutation with ensemble strategies. EARSDE exhibited very competitive results against most of the existing scalable benchmarks used in the literature on niching. This algorithm also for the first time uses arithmetic recombination operation in addition to the usual binomial/exponential crossover of DE in the context to multimodal optimization.

5.5 Large Scale Optimization

Large-Scale optimization refers to a special category of optimization problems where the number of decision variables, objectives and/or constraints can scale to extremely large values. Typically for real parameter EAs, locating the optima in a very high-dimensional (more than 100 dimensions) space can be very challenging. This is not surprising and is primarily caused by the exponential increase of the search volume with dimensions.

Usually the distance measures break down in higher dimensionalities and a search strategy that is valuable in small dimensions might be useless in large or even moderately high dimensional search spaces.

DE has been modified in different ways to search efficiently in a high-dimensional continuous space. Prior to 2010, Brest *et al*. (Brest *et al*., 2008) extended the adaptive jDE algorithm as jDEdynNP-F by gradually reducing the population size with generations and using a new scheme for changing the sign of F. Subsequently, Brest *et al*. (2010) proposed a variant of the jDEdynNP-F algorithm (Brest *et al*., 2008) by introducing two new features while retaining the prior framework. The first feature is performing a perturbation of the current best with randomly initialized vectors by DE/rand/1 strategy and a small scale factor lying in the range $[0.01, 0.09]$, only twice per generation. The second feature is the use of exponential crossover alongside the popular binomial with equal probability. Wang *et al*. (2010) extended the DE Enhanced by Neighborhood Search (DENS) (Yang *et al*., 2008) for large scale global optimization in the following way. The algorithm first generates a trial vector by either DE/rand/1/bin or by DE/rand/1/exp strategy with equal probability and replaces the target, if trial vector is fitter. A modified DE/target-to-best/1 with the archived previous best and two new randomly selected vectors from the population are used to construct a vector L. A similar strategy with the current best and the random vectors selected in the trial vector generation process are used to create another vector G. A second selection takes place with the previously selected target and the two newly created vector, where the fitter becomes a member of the evolved population.

Weber *et al*. (2011a) presented a sub-population based DE to adapt the algorithm for large scale optimization problems. The initial population is distributed with equal probability among some predefined number of subpopulations. At start, the subpopulation specific scale factor is initialized with values randomly taken from the range [0.1, 1], and mutation takes place following DE/rand/1/exp strategy, where the random vectors can be selected from all over the population. With a certain probability, a shuffling is performed by first uniting the subpopulations then recreating them by randomly redistributing the solutions. An update of the scale factor values for the sub populations are also performed after a generation with some probability. García-Martínez *et al*. (2011) defined a variation of roles played by the participating vectors in a regular DE mutation scheme. For example, the vectors picked for finding the difference vector, and the base vector plays different roles at the time of mutation. To achieve maximal gain from this differentiation scheme, the authors suggested selecting a vector only from the pool of those vectors in the current population that are best suited for performing the desired role. For example, the base vector will be selected among the top b vectors. In context to scaling DE with the dimensionality of search spaces, Zhao and Suganthan (2013) demonstrated the downsides to the commonly used exponential crossover in DE and proposed a linearly scalable exponential crossover operator (LS-EXP) based on a number of consecutive dimensions to crossover. Their numerical results indicate that LS-EXP exhibits a superior performance over the current exponential crossover operator on the most recent benchmark problems with dimensions ranging from 50 to 1000.

Omidvar *et al*. (2014) proposed a new technique to automatically decompose a problem based on the interdependencies of the decision variables. The decomposition scheme first identifies the interacting variables and then groups them based on common dependency (linkage) properties. A neighborhood-search based variant of DE known as SaNSDE (Yang *et al*. 2008) is used with this technique to optimize the subcomponents separately. Lopez *et al*. (2015) presented a hybrid of DE and the Variable Mesh Optimization (VMO) proposed by Puris *et al*. (2011). The DE is used to evolve and improve the quality of the initial mesh generated by the VMO operations, before utilizing it in the next iteration of the VMO. Very recently, Sayed *et al*. (2015) dealt with the problem of large scale constrained optimization and proposed a new technique namely, Variable

Interaction Identification for Constrained problems (VIIC) to detect the epistasis or dependence among the variables. The idea behind VIIC is to identify the groups of dependent variables such that they can be optimized separately. The authors integrated this scheme with DE and reported competitive performance on standard large scale test problems with constraints.

5.6 Surrogate-assisted DE for Expensive Optimization Problems

In real life, optimization problems may often become very much expensive in the sense that they can require an enormous amount of Function Evaluations (FEs) to reach an acceptable solution and/or each evaluation of the cost function may take very long time. Canonical EAs cannot directly solve them since the huge computational burden is unaffordable. As a remedy, surrogate-assisted evolutionary computation uses efficient computational models, often known as surrogates or meta-models, for approximating the fitness function in EAs (Jin, 2011). Although very small in number, some research efforts have been recently reported about the use DE for surrogate-assisted expensive optimization. Before 2010, such efforts of using DE in a surrogate assisted framework were not significant. Since the success of surrogate-assisted EAs depends not only on the construction and integration of the surrogate models but also on the efficiency of the underlying optimizer, due to its competitive performance, DE can stand out as a good choice for the core optimizer in such cases.

Lu *et al*. (2011) pointed out that each offspring in DE is only compared to the corresponding parent rather than with any other individual. Hence, neither the fitness values nor the ranks of offspring individuals are required by the algorithm. Thus, only a surrogate that can detect the better one between a parent and its offspring should serve the purpose. Driven by this intuition, the authors proposed the construction of surrogate for DE as a classification problem rather than a regression or ranking problem. For each parent in current population, a training set is chosen from the historical evaluated individuals and then a classifier is built on the training set. After this, the classifier is used to judge whether its offspring individual is better than itself so as to decide whether to evaluate the offspring individual with the fitness function. Evidently with this scheme, all offspring individuals are not required to be evaluated with the fitness function and in each generation, some FEs are saved for DE.

Elasayad *et al*. (2014) presented a surrogate assisted DE scheme in which a kriging (based on spatial linear regression and Gaussian correlation models to estimate the shape of the objective function from data) model is used to approximate the objective function while DE employs a mechanism to dynamically select the best performing combinations of parameters (F, Cr and Np). Miruna and Baskar (2015) integrated a diversity controlled and parameter adapted DE with local search into two dynamic surrogate models based on artificial neural networks and response surface methodology. The resulting algorithms were found to significantly reduce the required number of FEs for complex and multimodal problems without sacrificing the success rate.

Mallipeddi *et al*. (2015) presented an evolving surrogate model-based differential evolution (ESMDE) method, wherein a surrogate model constructed based on the population members of the current generation was used to assist the DE algorithm in order to generate competitive offspring using the appropriate parameter setting during different stages of the evolution. As the population evolves over generations, the surrogate model also evolves over the iterations and better represents the basin of search by the DE algorithm.

A different aspect in looking at the surrogate assisted DE is that the optimizer's computation results need to be available very quickly despite the long time required for each function evaluation. In this context, Glotić and Zamuda (2015) recently proposed a surrogate DE that was run over power challenges (hydrothermal

optimization), combining parallelization, kriging/surrogate matrix pre-computed model, and surrogate-assisted (mixed) handling of constraints.

Even though surrogate methods have been investigated for decades, MVMO (Rueda and Erlich 2015) method without any surrogate was the winner at CEC 2014 and CEC 2015 competitions on expensive optimization problems (Chen *et al*. 2014).

6. Hybrid DE Algorithms

For over the years, researchers have tried to combine the algorithmic components of DE with those of other metaheuristics like Particle Swarm Optimization (PSO), GA, Artificial Bee Colony (ABC), Covariance Matrix Adaptation Evolution Strategies (CMA-ES) and so on, with a belief that such hybrid algorithms will benefit from the synergy. An extensive account of hybrid algorithms developed in the broad perspective of computational intelligence can be found in Brest *et al.* (2015). It has been pointed out in (Blum *et al*., 2011) - "In fact, choosing an adequate combination of complementary algorithmic concepts can be the key for achieving top performance in solving many hard optimization problems". Several local search methods have also been integrated into the DE framework to enhance the capability of detailed search in promising regions. This led to the development of several memetic DE variants. In this section, we review the latest developments in hybrid DE in two parts: synergy between DE and other global search methods and the blending of DE with local search algorithms (memetic DE).

6.1 Hybridization of DE with other Global Optimizers

Probably DE has been most often hybridized with PSO, a simple continuous parameter optimizer inspired by the dynamics of social insect groups like school of fish or flock of birds. One reason behind this trend may be both the algorithms use simple difference operations to perturb the current solution. In fact the difference between the best and the current individual is explicitly used both in the velocity update scheme of PSO and in DE/current-to-best/1 mutation scheme. A comprehensive review and taxonomy of the plethora of PSO and DE hybrids can be found in (Xin *et al*., 2012). Some recent and notable attempts to hybridize DE and PSO can be found in (Epitropakis *et al*., 2012; Miranda and Alves, 2013). Elsayed *et. al* (Elsayed 2011b) hybridized real coded GA with differential operators of DE to solve CEC 2011 competition problems. This hybridization performed the best on the competition problems. Recently Trivedi *et al*. (2015) proposed a hybrid of DE and GA to solve the unit commitment scheduling problems. GA was used to handle the binary unit commitment variables whereas DE was employed to optimize the continuous power dispatch related variables. Liao (2010) proposed two hybrid variants of DE, one with a local search to improve exploitation capability, and the other with the music inspired Harmony Search (HS) algorithm to cooperatively find the global optimum. To handle the situation of mixed discrete and real valued dimensional problems, the author suggested a generalized discrete value handling based on the idea of table look up. Some other notable synergies between DE and HS have recently been reported in (Dash *et al*., 2014; Gao *et al*., 2014). Boussaïd *et al*. (2011) used a hybrid of DE and Biogeography Based Optimization (BBO) to provide a numerical solution to the optimal power allocation scheme in a wireless sensor network.

Olenšek *et al*. (2011) presented a global search algorithm, hybridizing DE and Simulated Annealing (SA). The algorithm can run in parallel following asynchronous master-slave architecture. The serial searching of SA is replaced by a population of size *Np*, which is subsequently modified following steps of DE instead of the random sampling of SA. The authors also used a local search to provide faster convergence. Another interesting hybrid of DE and SA has been reported in (Guo *et al*., 2014). Ghosh *et al.* (2012a) proposed a new hybrid

algorithm by embedding the DE type mutation, crossover, and selection operators in the framework of CMA-ES, with a view of improving the performance of the latter on noisy and complicated fitness landscapes (like hybrid composition functions of the CEC 2005 test-bed). Very recently, Li *et al*. (2015) studied different ways of population generation by DE and CMA-ES. The advantage of DE style generation is the nonparametric distributed search space coverage, with a direct parent offspring relation in successive generations. On the other hand CMA-ES style generation uses a parametric intermediate distribution to generate offspring and modifies the distribution with cumulative information gathered from all the previous generations. The authors proposed a new hybrid algorithm based on the DE framework that will also benefit from the addition of key features of CMA-ES. The algorithm generates a trial vector by first using a DE/rand/1/bin strategy followed by an Evolution Path (EP) mutation of CMA-ES.

Pholdee and Bureerat (2013) proposed a hybrid algorithm by incorporating the trial vector generation scheme of DE in a gradient based Real-Coded Population-Based Incremental Learning (RCPBIL) algorithm (Bureerat, 2011), with a view of overcoming the individual shortcoming of the two strategies. The authors described an extension of RCPBIL for multi-objective optimization, by making small modification to handle Pareto optimality. A few prominent examples of DE hybridized with other global optimizers are provided in Table 2.

6.2 Hybridization of DE with Local Search Methods: Memetic DE

Reynoso-Meza *et al*. (2011) presented a hybrid algorithm, where regular DE is combined with a sequential quadratic programming type local search. The algorithm also uses a novel parameter adaptation technique, where triangular distribution is used to generate the control parameters. Jia *et al*. (2011) proposed a memetic DE that comes with a chaotic local search, which shrinks its search space over generations, to preserve and improve the solution quality. Neri *et al*. (2011) proposed a compact DE, hybridized with a memetic search to achieve faster convergence. The algorithm represents the population with a multi-dimensional Gaussian distribution. Zhan and Zhang (2012) modified the binomial crossover scheme of DE to allow a dimension of the trial vector to perform a random walk, with a certain probability. The resulting DE algorithm could achieve better performance over classical DE schemes especially on multimodal problems.

Poikolainen and Neri (2013) proposed a hybrid DE variant, where a Hooke-Jeeves method based local search has been used alongside regular DE. In this algorithm, at first after selecting *Np* number of solutions uniformly from the search space, the local search (with a small number of iterations) is used for a primary refinement, to form the initial population. DE/rand/1/exp scheme is then used to generate the population for the next generation, and *Cr* is set in such a way that in average it can copy n_e number of genes from the donor. *Zhang et al*. (2013) presented a distributed DE, which uses a local search to fine tune the solutions and migrations of solutions among subpopulations are controlled by Lamarckian and Baldwinian learning. The distributed DE uses *m* number of subpopulations, each of size *Np/m*. The subpopulations are arranged in Von Neumann topology as described by (Dorronsoro *et al*., 2011). To fine tune the solutions and avoid the case of stagnation, the authors proposed the use of Hooke–Jeeves local search algorithm. A new memetic DE was presented by Piotrowski (2013), where a local search algorithm has been introduced into the framework of the DEGL (DE with Global and Local neighborhoods) algorithm (Das *et al*., 2009). For local search the authors proposed the use of Nelder-Mead algorithm (Nelder and Mead, 1965), which can be employed with a long search length, on the best individual in a neighborhood formed by a percentage of the population.

Methods Hybridized	Authors	Optimization problems applied to
DE + Artificial Bee Colony (ABC)	(Tran *et al*., 2015)	Multi-objective optimization for optimal time-cost-

Algorithm		quality trade-off for planning of construction projects
DE + Ant Colony Optimization (ACO)	(Chang *et al*., 2012)	Single-objective real parameter engineering optimization
DE + Bacterial Foraging based Optimization (BFO)	(Biswal *et al*., 2012)	Single-objective continuous parameter clustering problem
DE + Gravitational Search Algorithm (GSA)	(Chakraborti *et al*., 2015)	Binary optimization problem involving highly discriminative feature subset selection
DE + Invasive Weed Optimization (IWO)	(Basak *et al*., 2013a)	Single-objective, bound constrained function optimization problems
DE + Firefly Algorithm (FFA)	(Abdullah *et al*., 2013)	Single-objective nonlinear optimization problem involving estimation of biological model parameters
DE + Fireworks Algorithm (FWA)	(Zheng *et al*., 2015)	Single-objective, bound constrained function optimization problems

Table 2: Examples of recently developed hybrid DE algorithms

An adaptive memetic search algorithm was proposed by Rakshit *et al*. (2013), where a hybrid of DE and local reinforcement learning based refiner namely, Temporal Difference Q Learning (TQDL) is used. The basic idea is to reward a performing parameter while penalizing an unsuccessful one, following the rules of TQDL learning and the Q table. DE/current-to-best/1 mutation scheme is used and the scaling factors for each individual are adapted based on the reward/penalty given by the learning algorithm. The algorithm was successfully used in an application involving multi-robot path planning. Qu *et al.* (2014) further improved the fitness-Euclidian distance ratio (FER) DE proposed by Liang *et al.* (2014). The authors pointed out that the previous algorithm will favor those solutions, which have a high FER. This unfortunately may end up giving higher probability of selection to an individual situated far from the target, hindering convergence. On another note, the problem of finding global optimum is less complex and requires less diversity of population than a multimodal optimization problem (for which the previous algorithm is designed). These observations lead to a modified version of FER that uses a normalization operator. A Quasi-Newton method based local search is used here to further fine tune the best solution found, if possible.

Hybridizing DE with other search methods has been a popular research area. A common justification used by researchers is stated as "No other researcher has hybridized DE with this specific search method". As numerous new names are created incessantly, there can possibly be a large number of hybridized algorithms. An appropriate objective to be used in the future research for hybridization should be demonstrating improved performance by the hybridized algorithm over the best state-of-the-art of the individual algorithms.

7. DE for Discrete and Combinatorial Optimization Problems

DE is by nature a continuous parameter optimizer which directly searches on subsets of $\mathbb{R}^d$. However, there have been several attempts to modify and use DE for binary and combinatorial optimization problems. Several such works have been discussed in our earlier 2011 DE survey. Here, we outline some recent approaches in this direction.

Wang *et al*. (2012) proposed a binary DE that uses a randomly generated initial binary population of size *Np*. The problem of performing mutation on binary strings is solved by producing a probabilistic model of the mutation, and then generating a donor following the model. The probability model for the j^{th} dimension of the i^{th} target is defined as follows:

$$p_{i,j} = \frac{1}{1 + e^{-2b(MO-0.5)/1+2F}},$$

where $MO = \mathbf{x}_{R_1 i}{}^{(t)} + F\left(\mathbf{x}_{R_2 i}{}^{(t)} - \mathbf{x}_{R_3 i}{}^{(t)}\right)$. Now, the donor is generated as follows:

$$v_{i,j} = \begin{cases} 1 \; if \; rand(0,1) < p_{i,j}, \\ 0 \; otherwise. \end{cases}$$

Crossover and selection can be directly adopted from general DE as they do not depend on the numerical properties of the solution. The key feature of the algorithm is, due to the probabilistic modeling of the mutation, it is not affected by the constraint of the population being binary in nature, and thus can adopt and benefited from any variant of DE. This strategy has been recently applied to devise a hierarchical binary DE for solving the minimal cut-sets identification problem in (Maio *et al*., 2015)

In most of the occasions when DE needs to operate on discrete valued decision variables, a usual practice is to convert a real-valued solution into a desired integer-valued solution by applying some posterior decoding mechanisms. Recently Dutta and Figueira (2013) proposed a DE algorithm that can directly work with real, integer, or discrete variables without the need of such conversion. The proposed DE uses a binary representation of integers and let *b* be the minimum number of bits to represent any integer in the search space. The solution is represented as a string of the concatenated binary representation of the values in each dimension i.e. a *bd* dimensional vector, where *d* is the dimension of the problem. The mutation technique is done by first performing a regular mutation on the *bd* dimensional solution. The donor vector can become a *bd* dimensional real valued vector, due to mutation, and need to be discretized. If an element of the donor vector is in the range [0, 1] it is transformed to 1, else it is 0. But, this approach uses a fixed transformation, which may not be suitable to maintain the randomized environment of DE, may lead to local optima and dimensionality is increased thereby adversely affecting performance on large scale problems. Thus, the authors suggested that for each of the element of the donor, either the above mentioned transformation or its reverse will take place with a certain probability.

Economic Lot Scheduling (ELS) under basic period policy amounts to finding the best cyclic production schedule of (say) *n* items to be produced on a single machine such that the production cycle of each item is an integer multiple of the basic period. The process assumes deterministic demand and production rates that are known a priori. Tasgetiren *et al*. (2011) proposed a DE based heuristic approach to solve such an ELS problem. The authors illustrated competitive results against the existing GA based approaches.

Deng and Gu (2012) used a hybrid discrete DE with neighborhood search to solve no-wait flowshop scheduling problems with makespan criteria. Tasgetiren *et al*. (2013) integrated a variable iterated greedy search with DE to solve the no-wait permutation flow-shop scheduling problems. Recently Maravilha *et al*. (2014) presented a combinatorial optimization framework based on DE. In this algorithm a set-based representation and operators are used to define sub-problems that are then utilized to explore the feasible search volume. Authors tested the proposed method on the capacitated centered clustering problem.

A first approach to use DE for solving the multi-dimensional knapsack problem was developed by Tasgetiren *et al*. (2015). The authors employed a variable neighborhood search in conjunction with different mutation strategies of DE to generate the trial population. Although the proposed algorithm works on a continuous domain, these real-values are converted to 0-1 binary values by using the sigmoid function. In order to enhance the solution quality, DE with the variable neighborhood search was combined with a binary swap local search technique.

Chen *et al*. (2015) proposed a new DE algorithm that can optimize binary coded problems. The algorithm maintains two populations of size *Np* initialized with same random solutions. After the first generation one population is used as the current and the other as an archive for the solutions of the previous generation. However, the algorithm didn't follow a signature DE scaled difference vector based mutation, or didn't even

apply a crossover. The suggested mutation scheme is more similar to the one, done in PSO, though some measures are taken to reduce the pre mature convergence or stagnation common to such strategies.

8. Theoretical Analysis of DE

Theoretical analysis of EAs is very much important to understand their search mechanisms, to detect the allowable ranges of the control parameters, to find problem classes in which the algorithm with a given set of parameters will perform successfully (with some quantification of success) or will fail. Unfortunately, unlike PSO or ESs (Evolutionary Strategies) the framework of DE does not easily lend itself to analytical treatments, unless some simplifying (and often impractical) assumptions are made. In this section, we provide a brief overview of the few theoretical studies undertaken mainly over the last five years on DE as a function optimizer.

Kitayama *et al*. (2011) tried to identify the basic properties required by the search techniques of DE. They pointed out that a direction of searching, which will favor a solution improving local minimum, but at the same time will discourage any that will not improve the global status, is useful for escaping from local trough. They also supported the inclusion of randomness in the search technique, which favors the opportunity of exploration in an unknown environment. By illustrative examples they showed how the rand/1/bin strategy actually follows the above mentioned properties and provides opportunity for a solution to escape from a local minimum and explore the neighborhood, maintaining diversity.

In an attempt to establish the theoretical basis of the DE, Ghosh *et al*. (2012b) provided a proof of the convergence of a DE algorithm following the DE/rand/1/bin strategy. The proof is provided only for those optimization problems whose objective functions are continuous and posses a single global optimum (may contain a number of local optimums), and there exists, continuous first and second order derivatives. The authors modeled the DE population with a random vector and showed that its probability distribution function (PDF) reaches to an equilibrium PDF when the number of generation reaches infinity. The equilibrium PDF is shown to be similar to a Dirac delta function that has a zero probability of finding a fitter point only the global minimum. For any other point in the search space the equilibrium PDF will have a non-zero probability of finding a better point, thus confirming that the DE will be able to find the global minimum. The authors also established the fact that there is a Lyapunov functional associated with the DE system. This functional can be expressed as the difference between the mean fitness of the population and the optimal fitness. The functional is shown to be monotonically decreasing over time indicating at convergence the mean fitness will reach to the optimum fitness or the population will improve over time to converge in the neighborhood of the global optimum. The importance of the difference vector mutation scheme is also established by the authors.

Zhao *et al*. (2009) introduced a hybrid DE algorithm (HtDE) based on the concept of the transform functions and proved the convergence of the same under some restrictive assumptions. He *et al*. (2010) used the so called Differential Operator (DO) to obtain a random mapping from the decision variable space to the Cartesian product of the former and subsequently investigated the asymptotic convergence of DE by using the random contraction mapping theorem. A Markov chain modeling of DE was developed by Sun (2009) who inferred that classical DE does not show convergence in probability in its usual sense.

Recently, Hu *et al*. (2014) derived two sufficient conditions that can assure the convergence of DE in the usual sense. According to the derived conditions, the DE-variants can guarantee convergence to a globally optimal point provided the probability of generating an actual optimum (or optima) by the reproduction operators in each generation in a certain sub-sequence of the population remains greater than a small positive number. The fundamental problem with this approach is that they considered the distribution of the population in each iteration to be independent of each other, which is generally not the case, as any population in a

particular iteration is completely determined by the previous iteration. Hence, even intuitively it is not acceptable that the probability distribution of the population in each iteration will be independent of the distributions in earlier iterations.

9. Parallel DE Algorithms

Since the last decade of 21st Century, parallel computing emerged as a form of high-performance computing owing to the dramatic cost reduction and abundance of computational resources (both software and hardware). In parallel computing, several calculations can be undertaken simultaneously based on the principle that a large problem can be divided into smaller ones which can then be solved concurrently. Like many other EAs, DE has also been parallelized for enhancing its speed and accuracy on expensive optimization problems. Discussions on some recent and interesting studies on parallel DE algorithms are in order.

Wang *et al*. (2013) proposed a parallel DE scheme by using an adaptive parameter control and Generalized Opposition based Learning (GOBL) (Wang *et al*., 2011), which will be useful for high dimensional optimization problems. This variant can also be implemented in a parallel processing environment, for example in a graphical processing unit (GPU), which will provide a massively large and fast computational power. GOBL for every solution creates an opposite solution and also it retains a dynamic range of the dimensions of the population, such that the knowledge of the shrinking search space with generation can be kept in record. The authors adopted the parameter adaptation scheme from jDE (Brest *et al*., 2006), and only modified the allowable range of *Cr*, changing it to [0.8, 1]. The proposed algorithm will either apply GOBL or classical DE with a probability. In GOBL, after updating the dynamic range of solution, opposite solutions will be generated to form another population. The *Np* best individuals will be selected from the union of the current and the opposite population, to form the population for the next generation. While the above algorithm is for executing in CPU, the authors also presented an implementation scheme, by defining the kernel functions, to execute it in a GPU.

Distributed DE enhances the exploration power of DE by performing an organized and landscape specific customized search. Being one of the major impactful control parameter for common DE, the scale factor in distributed DE is far more important, and complicated to choose and adapt. The reason is the requirement for different adaptation strategies and choice of the scale factor, for different local landscapes. Weber *et al*. (2011b) proposed to use four variant of scale factor adaptation schemes, in different sub populations to tackle the issue. The first scheme is to select a scale factor uniformly from the range [0, 1], for each subpopulation. The scale factor remains constant throughout the run of the algorithm. In the second scheme, each k^{th} subpopulation takes a diverse scale factor in the following way:

$$F_k = \frac{k-1}{m} + \frac{1}{2m},$$

where *m* is the number of subpopulations. For the second scheme also the scale factor is not adaptive, but allows the subpopulation to evolve in a diverse manner. The third scheme picks a scale factor randomly from the range [0, 1]. But, unlike scheme one; it revises its decision by checking the improvement of every subpopulation after a certain interval. The scale factor of the sub population, which is found to be least improved, is replaced with another random scale factor. In the fourth scheme, each subpopulation is assigned a random scale factor, and in each generation, one randomly selected subpopulation gets its scale factor revised with another randomly chosen one. The authors presented an analysis using three distributed DE algorithms, Parallel DE (PDE) (Tasoulis *et al*., 2004), Island based Distributed DE (IBDDE) (Apolloni *et al*., 2008), and the Distributed DE (DDE) proposed by Falco *et al*. (2007) and validated the importance of the proposed scale factor choosing strategies. The obtained results through various experiments, showed firstly an algorithm dependence of schemes'

performance, which is natural and can be explained by the different strategy of migration of solutions between subpopulations, and disparate nature of the subpopulations (scheme two, three and four performed better in PDE, ISDDE and DDE, respectively), secondly PDE with scheme three provides a globally good performance, establishing the importance of adaptation.

The influence of migration on the performance of DE was empirically examined by Bujok and Tvrdík (2012) by applying six adaptive DE-variants to a parallel migration model with a star topology. Chen *et al*. (2013) developed a parallel DE scheme for optimizing large-scale atomic and molecular clusters based on the sum of pair-wise potential minimization. This approach combines a modified DE algorithm with improved genetic operators and a parallel strategy with a migration operator to address the problems of numerous local optima and large computational demanding. Penas *et al*. (2014) used an asynchronous parallel implementation of DE for the parameter estimation in dynamic models of biological systems.

There have been some important attempts to implement DE and its variants on modern parallel computing platforms based on super parallel SIMD (single-instruction multiple-data) devices like GPUs. A first parallel implementation of DE on a GPU with NVIDIA's compute unified device architecture (CUDA) was by Krömer *et al*. (2013). Wong *et al*. (2014) recently reported a CUDA based parallel implementation of the well known SaDE (Qin *et al*., 2009) algorithm. Experimental results provided by the authors indicate that such parallelization can significantly speed up SaDE as compared to its single processor sequential version across varying problem dimensions.

9. Engineering Applications of DE

It is needless to mention that with the growing popularity of DE among the practitioners, parallel to the core algorithmic research in DE, the application specific research on and with DE has also been spiking over the last 5 years. DE research articles indexed in SCI database over the span of 2011– 2015 is 4750 and out of these, there are more than thousands of application papers in diverse areas. Instead of going into any detailed discussions (which is also out of the scope of a single survey, given the volume of works published), in Tables 3 - 5 we highlight only the major applications, where DE has been employed to solve the optimization problem, along with the type of the DE used. Please note that to keep the reference list tractable, all references mentioned in Tables 3 – 5 are kept in the supplementary document named "DE Application References".

Table 3: Summary of Applications of DE to Engineering Optimization Problems

Electrical and Power Systems	
Economic Dispatch	Bhattacharya and Chattopadhyay (2011): Hybrid of DE and biogeography based optimization. Sayah and Hamouda (2011): Hybrid of PSO and DE. Basu (2011): Multi-objective DE. Ghasemi *et al*. (2014): Hybrid of teaching learning algorithm and a variant of DE, with double mutation and crossover. Pandit *et al*. (2015): Problem dependent mutation strategy (DE/rand/1, DE/best/1, DE/rand-to-best/1, DE/rand/2, DE/best/2) and binomial crossover. Glotić and Zamuda (2015): surrogate assisted DE variants combined in a master-slave optimization model. Mallipeddi *et al*. (2012): ECHT-DE
Power System Stabilizer	Vakula and Sudha (2012): DE/rand/1/bin.
Fault Diagnosis	Zhao *et al*. (2014): A variant of DE called history driven DE.
Power Distribution Reconfiguration	Prado *et al*. (2014): Discrete DE.
Optimal Power Flow	Sivasubramani and Swarup (2012): DE/rand/1/bin.
Artificial Neural Networks	
Optimal Network Topology	Dragoi *et al*. (2013): Self adaptive DE.

Designing of Different Variants of Neural Networks	Aliev *et al*. (2011): DE/rand/1/bin. Dhahri *et al*. (2012): A variant of DE, named as hierarchical multi-dimensional DE. Oh *et al*. (2012): DE/rand/1/bin.
Neural Network Training	Subudhi and Jena (2011b): Hybrid of DE and local search. Piotrowski (2014): Eight popular variants of DE (DEGL, JADE etc).
Fuzzy Neural Net Controller	Lu *et al.* (2012): DE with modified mutation and binomial crossover.
Neural Network for Non linear System Identification	Subudhi and Jena (2011a): Opposition based DE with DE/rand/1/bin scheme for trial generation.
Manufacturing Science and Operation Research	
Manufacturing Process Optimization	Zhang *et al*. (2013): Hybrid of DE and Tissue *P* Systems. Yildiz (2013b): Hybrid of DE and receptor editing property of immune system.
Transport Sequencing in Cross Docking Systems	Liao *et al*. (2012): Two hybrid DE variants.
Optimization of Multi-Pass Turning Operations	Yildiz (2013a): Hybrid of DE and Taguchi's method.
Scheduling	Ponsich and Coello (2013): Hybrid of DE and Tabu Search. Zhang *et al*. (2013): Hybrid of DE and local search. Pan *et al*. (2011): Hybrid of discrete DE and local search. Vincent and Ponnambalam (2013): Bi-level DE. Tang *et al*. (2014): DE with a modified DE/current-to-best/1 mutation. Glotić *et al*. (2014): Self-adaptive DE.
Ship route planning and marine safety	Zhao *et al*. (2014): Improved variant of DEGL
Robotics and Expert Systems	
Space Trajectory Optimization	Vasile *et al*. (2011): A variant of DE known as inflationary DE.
Route Planning/Guidance of Unmanned Vehicles	Fu *et al*. (2013): Hybrid of DE and quantum behaved PSO. Raghunathan and Ghose (2014): DE/rand/1/bin, DE/best/1/bin. Zamuda and Sosa (2014a): DE/best/1/bin with adaptive parameter settings. Zhang and Duan (2014): DE with α level comparison for constraint handling.
Satellite Orbit Reconfiguration	Chen *et al*. (2015): A variant of multi-objective DE.
Part based Work Piece Detection	Liu *et al*. (2012): DE with rand/1 mutation and crossover.
Real Time Object Tracking	Nyirarugira and Kim (2013): DE with modified mutation and crossover.
Object Detection	Ugolotti *et al*. (2013a): DE/rand/1/bin with GPU implementation.
Body Pose Estimation	Ugolotti and Cagnoni (2013b): DE/rand/bin in GPU.
Financial Market Dynamics	Hachicha *et al*. (2011): DE with modified rand/1 mutation scheme and binomial crossover.
Neural Fuzzy Inference System	Chen and Yang (2014): Depending on requirement uses five mutation schemes and binomial crossovers.

Table 4: Summary of Applications of DE to Engineering Optimization Problems (Continued from Table 3)

Pattern Recognition	
Classification	Qasem and Shamsuddin (2011): Hybrid of multi-objective DE with memetic search and radial basis neural network. Luukka and Lampinen (2011): DE/rand/1/bin. Triguero *et al*. (2011): DE, SADE, JADE, DEGL, SFLSDE (Neri and Tirronen 2009). Triguero *et al*. (2012): DE/rand-to-best/1/bin. Bazi *et al*. (2014): DE/rand/1/bin. Zhai and Jiang (2015): Hybrid of self-adaptive PSO and DE.
Clustering	Zhong *et al*. (2013): Multi objective DE. Dong *et al*. (2014): Adaptive DE with multiple strategies. Kwedlo (2013): DE/rand/1/bin.
Feature Selection	Khushaba *et al*. (2011): DE/rand/1/bin. Al-Ani *et al*. (2013): Modified DE/rand/1/bin. Paul and Das (2015): MOEA/D-DE.
Image Processing	
Moving Object Detection	Ghosh *et al*. (2014): distributed DE with neighborhood based mutation.
Image Segmentation	Novo *et al*. (2013): Hybrid SPEA-2 (Strength Pareto Evolutionary Algorithm) and DE.
Multi-level Image Thresholding	Ali *et al*. (2014): DE/rand/1/bin. Sarkar *et al*. (2015): DE/rand/1/bin.
Feature Selection in Image Data	Ghosh *et al*. (2013): Self-adaptive DE.

Sub-pixel Mapping	Zhong and Zhang (2012): DE/rand/1/bin with adaptive control parameter values.
Animated Tree Reconstruction	Zamuda and Brest (2014b): DE/rand/1/bin with adaptive parameter settings. Zamuda *et al*. (2011): DE/rand/1/bin with adaptive parameter settings.
3D Reconstruction from Uncalibrated Images	Kang *et al*. (2013): DE/rand/1/exp.
Bioinformatics and Bio-medical Engineering	
Hypoglycaemia Detection	Lai *et al*. (2013): Multi-objective DE, with double wavelet mutation.
Rule Extraction from Medical Database	Falco (2013): DE with multiple strategies defined by Price *et al*. (2005).
Hippocampus localization in histological images	Mesejo *et al*. (2013): DE/target-to-best/1/bin.
Monitoring of obstructive sleep apnea	Sannino *et al*. (2014): DE/rand-to-best/1/bin.
Parameter Estimation of Biological Systems	Zhan *et al*. (2014): Modified DE/rand/1/bin.
Electronics and Communication Engineering	
Mobile Location Management	Almeida-Luz *et al*. (2011): DE/best/1/exp.
Non-linear System Modeling	Chang (2012): DE/rand/1/bin.
Clustering of Wireless Sensory Network	Kuila and Jana (2014): DE/best/1/bin.
Mobile Ad-hoc Networks	Gundry *et al*. (2015): DE/rand/1/bin.
Optimal Fault Protection in Networks	Li *et al*. (2013): Binary multi-objective DE. Zio *et al*. (2012): A variant of binary DE.
Placement of Wavelength Convertors.	Lezama *et al*. (2012): Binary DE with DE/rand/1/bin trial vector generation strategy.
Sleep-scheduling in wireless sensor networks	Sengupta *et al*. (2012): MOEA/D-DE
Electromagnetics including antenna design	Rocca *et al*. (2011): A survey on the use of DE in Electromagnetics. Secmen *et al*. (2013): Ensemble DE Baatar *et al*. (2014): Multi-objective DE.

Table 5: Summary of Applications of DE to Engineering Optimization Problems (Continued from Table 4)

Miscellaneous	
Speech Processing	Schleusing *et al*. (2013): DE/rand/1/bin. Lei *et al*. (2013): DE/rand/1/bin.
Filter design	Ghosh *et al*. (2012): Modified JADE with *p*BX crossover
Parameter Estimation in Systems	Banerjee and Abu-Mahfouz (2014): DE/rand/1/bin, DE/best/1/bin, DE/target-to-best/1/bin. Tang *et al*. (2012): DE/rand/1/bin. Tsai *et al*. (2011): DE with Taguchi sliding level method based crossover.
Beamforming in Signal Processing	Mallipeddi *et al*. (2011a, 2011b, 2011c)
Infinite Impulse Response System in Signal processing	Upadhay *et al*. (2014): DE with wavelet mutation. Zhu *et al*. (2012): Different popular variants of DE (SADE, jDE etc).
Optimization of Low-loss Silver Nano-wires Structure	Zhao *et al*. (2012): DE/rand-to-best/1/exp.
Fractional Order Systems	Zhu *et al*. (2012): A variant of DE known as switching DE.
Nuclear Plant Safety	Zio and Viadana (2011): Multi-objective DE. Maio *et al*. (2014): Two-step hierarchical DE.
Strength of Heat Source	Parwani *et al*. (2013): Hybrid of DE and local search.
Service Optimization	Pop *et al*. (2011): Discrete DE.
Waveform Inversion	Gao *et al*. (2014): Modified cooperative co-evolutionary DE.
Self Potential Data (Geophysics)	Li and Yin (2012): DE/current-to-best/1/bin.
Adaptive Many Particle Quantum Metrology	Lovett *et al*. (2013): DE/rand/1 with crossover.
Parametric identification of seismic isolators	Quaranta *et al*. (2014): DE/best/1/bin.
Chemical Engineering	Sharma and Rangaiah (2013): Multi objective DE. Kumar *et al*. (2011): DE with rand/1 mutation and crossover. Silva *et al*. (2012): DE/rand/1/bin. Vakili *et al*. (2011): DE/best/1/bin.
Engineering Design and Modeling of Machines and Components	Coelho *et al*. (2013): Multi-objective DE. Joly *et al*. (2013): Multi-objective DE. Bhattacharya *et al*. (2013): DE with mutation as in Karaboga and Okdem (2004) and binomial crossover. Saruhan (2014): DE/rand/1/bin. Kranjcic and ˇStumberger (2014): DE/rand/1/bin.

	Tsai (2015): DE/rand/2 mutation with Taguchi sliding level method based crossover. Deb *et al*. (2014): DE/rand/1 with crossover and DEGL.

10. Potential Future Research Issues

Although during the last two decades, research on and with DE has reached an impressive state, there are still some interesting open problems and new application areas are continually emerging for the algorithm. Below, we unfold some important future directions of research in the area of DE:

1. Rotation invariance has been a challenge for DE. Covariance matrix based mutation and cross-over operations have been integrated in DE (Ghosh *et al*. 2012a, Wang *et al*. 2014). This approach suffers from the lack of scalability as it involves repeated inversion of matrix whose dimensionality is the same as the dimensionality of the problem. Arithmetic recombination is also a rotation invariant operator which does not suffer from the lack of scalability as much as matrix inversion. Recently, arithmetic recombination (Hui *et al*., 2015) has been employed to solve niching problems effectively. We expect similar exceptional performance by arithmetic recombination in other problem categories too.
2. Population topologies have been extensively investigated in the context of particle swarm optimization (PSO). Index-based topologies (Das *et al*. 2009) have been commonly used while Euclidean distance based topologies (Suganthan 1999, Qu *et al*. 2012, 2013, Hui *et al*. 2015) have not been frequently used. The objective of using topologies in PSO is to slow down the rapid convergence of the particles towards the current *gbest* position (Lynn *et al*. 2015). However, DE posses slower convergence properties primarily due to the randomized mutation operators (independent of the current *gbest*) and parent-offspring competition. While rapid premature convergence of PSO must be tackled, improving the convergence behaviour of DE can also be beneficial. In this context, Euclidean distance based neighbourhood has been demonstrated to be effective in the context of solving niching problems (Qu *et al*. 2012). Therefore, Euclidean distance based neighbourhood operations in other problem scenarios can be investigated.
3. DE has not demonstrated its superior performance on computationally expensive problems. MVMO optimization method (Rueda and Erlich, 2015) has been the winner at CEC 2014 and 2015 Competitions on Expensive problems. Hence, novel strategies need to be developed to deal with expensive problems more competitively by using DE.
4. Recently, population size adaptation has been demonstrated to yield improved performance. Naturally, we require a larger population to perform exploration of the search space at the early stage of the search while we require a smaller population to conduct fine search near the best regions at the end of the search process. Benefits of such a population reduction have been demonstrated in works like L-SHADE (Tanabe *et al*., 2014a). Further research is needed in the population size adaptation in single objective and other optimization scenarios.
5. Learning based approaches (Liang *et al*. 2014b) for DE need to be developed. Even though DE researchers test DE on a large collection of optimization problems with diverse characteristics, a practicing engineer is likely to tackle a single problem with minor variations repeatedly over a prolonged period of time (Suganthan, 2015). In this context, learning based approaches will be valuable to solve minor variants of the same problem repeatedly by DE (López-Ibáñez 2011, Smit 2010)

6. No free lunch theorem states that no single instantiation of an algorithm can outperform all others. This implies that it would be beneficial to have several tunable parameters in an algorithm so that numerous instantiations would be possible by setting tunable parameters to different values in order to solve problems with diverse characteristics. Even though this approach seems contradictory to the concept of robust optimizer, investigating the benefits of increasing control parameters of DE is a promising research direction.
7. Evolutionary algorithms are used to solve multimodal problems within a limited computational budget. If a problem is unimodal or if we have unlimited computational budget, we can identify better alternatives to evolutionary algorithms. In this context, controlling population diversity and convergence behaviour over the given computational budget is important. Hence, one of the promising research directions will be hybridizing ensemble methods with Euclidean distance based topologies (Suganthan, 2015) where exploitative ensemble can be used in the neighbourhood of the best solutions while explorative ensemble can be used for weaker solutions to generate their offspring.
8. Even before the advent of population-based EAs, the concept of convergence was prevalent for single point (only one candidate solution) based search methods. If the single point does not converge, there will be no solution. However, in the context of population based algorithms, the best scenario is that even after one population member discovers the global solution the other members to be well distributed in the search space. As we do not know the exact global solutions (unlike solving the benchmark problems) in practical scenarios, this is an ideal scenario to have within the specified computational budget. Hence, in the context of population-based search methods, the focus of theoretical research can be controlling diversity and convergence behaviours while avoiding chaotic search behaviour within the given computational budget (Suganthan, 2015).
9. Expected First Hitting Time (EFHT) is the average time that an EA requires to find an optimal solution for the first time and it stands out as an important issue related to the theoretical investigation of the EAs. If the actual optimum is not known in practice, EFHT should consider the average time required to minimize the objective function below a predefined threshold. To the best of our knowledge, general development of computational complexity of DE is not available yet. The convergence rate and EFHT, which are the measures of average computational complexity of any optimization technique, are yet to be developed for the DE family of algorithms. The necessary and sufficient conditions for convergence, the dynamics of the DE population and the time complexity of the DE algorithm are interconnected issues that constitute the three main aspects of this optimization technique. Hence a unified approach of addressing these three issues simultaneously is a must to explore their inter-correlation. To the best of our knowledge, such a unified formulation is missing in the theoretical development of DE.
10. Memetic algorithms have been widely studied in the evolutionary computation domain. However, in the CEC competitions from 2005 to 2015, there is no compelling evidence of improved performance by memetic DE in any of these competitions. Further, integrating heterogeneous population topologies with exploitative and explorative ensemble configurations might be a better alternative than memetic DE (Suganthan, 2015). In this context, extensive comparative studies will be valuable to compare memetic DE and DE with heterogeneous population topologies with exploitative and explorative ensemble configurations in the context of numerical optimization.
11. In the paradigm of evolutionary computation, there is a general lack of a clear mapping between problem features and the best suited optimization algorithm (out of the plethora of choices among DE,

PSO, ABC, GA, CMA-ES and so on). This is especially true for solving real world optimization problems. A systematic study is required to determine which features of a particular problem (like multi-modality, separability, ruggedness of the functional landscape etc.) or what kind of correlation among a set of decision variables make an objective function solvable by DE (or any of its particular variants). As was also pointed out in (Das and Suganthan, 2011), almost eight years back from now, Langdon and Poli (2007) made a very interesting study where certain fitness landscapes were evolved with Genetic Programming (GP) to indicate the benefits and weaknesses of a few population-based algorithms like PSO, DE, and CMA-ES. The authors indicated that some problem landscapes found with GP may deceive DE such that it will be frequently stuck in local optima. On the other hand, over similar landscapes PSO was demonstrated to always find the global optima correctly within a given computational budget. Unfortunately this kind of studies was not undertaken on a larger scale to identify the suitable problem features for the vast class of DE variants available today. This is particularly challenging if real world problems possess diverse properties over the search space like the composition problems (Liang 2005).

12. Finally, while adapting the control parameters like *F* and *Cr*, many methods introduce controlled amount of randomness. It is certainly interesting to investigate when it is useful to increase or decrease the degree of randomization and which are the appropriate methods to do this. Can we link the amount of randomness with some features of the function to be optimized or on some kind of correlation among the decision variables? This can be a potential future avenue of research.

Hence, it is apparent that there are numerous issues to be investigated in the context of differential evolution. Further, some of the above issues can be investigated collectively as well as in different optimization scenarios such as constrained, multi-objective, dynamic, multimodal and so on.

11. Conclusions

Dramatic advances in hardware have paved the paths for solving very complicated computational problems by using computer-based optimization models. Such problem instances very often call for robust, fast and reliable optimizers. The DE family of algorithms initially meant for global numerical optimization over the continuous search spaces emerged about two decades ago. The field has been greatly enriched by the continual efforts of a number of researchers from various domains. In this article, we made an effort to outline the multi-faceted state-of-the-art research on and with DE spanning mostly over the years 2011 to 2015.

The field of metaheuristics is now growing at a spectacular rate – countless new algorithms are being published in various venues almost every day. Inspirations for designing a nature-inspired optimizer are coming from diverse sources ranging from human beings to flu virus! Similarly researchers are exploring various ideas from the coveted realm of mathematical programming to stretch them into novel metaheuristics. However, unlike such mushrooming algorithms, most of which provide a single and somewhat narrowed search strategy due to their source of inspiration, whereas DE provides the user with a flexible set of offspring generation strategies with strong intuitive justifications behind the resulting search moves. Such flexibility has made DE a versatile and robust optimizer in widely differing and difficult optimization scenarios. We firmly believe that DE will continue to remain a vibrant and active field of multi-disciplinary research in the years to come.

References:

A. Abdullah, S. Deris, S. Anwar, and S. N. V. Arjunan. 2013. An evolutionary firefly algorithm for the estimation of nonlinear biological model parameters. *PLOS one*, vol. 8(3).

M. M. Ali. 2011. Differential evolution with generalized differentials. *Journal of Computational and Applied Mathematics*, vol. 235, pp: 2205-2216.

M. Ali, P Siarry, and M. Pant. 2012. An efficient differential evolution based algorithm for solving multi-objective optimization problems. *European Journal of Operational Research*, vol. 217, pp: 404 - 416.

M. Z. Ali, N. H. Awad, P. N. Suganthan. 2015. Multi-population differential evolution with balanced ensemble of mutation strategies for large-scale global optimization. *Appl. Soft Comput*. 33: 304-327.

M. Ameca-Alducin, E. Mezura-Montes and N. Cruz-Ramírez. 2014. Differential evolution with combined variants for dynamic constrained optimization. In proc. of *IEEE Congress on Evolutionary Computation (CEC),* July 6-11, 2014, Beijing, China.

J. Apolloni, G. Leguizamon, J. Garcia-Nieto and E. Alba. 2008. Island based distributed differential evolution: an experimental study on hybrid test beds. In Proceedings of *IEEE International Conference on Hybrid Intelligent Systems*, pp: 696–701.

M. Asafuddoula, T. Ray, and R. Sarker. 2011. An adaptive differential evolution algorithm and its performance on real world optimization problems. In proceedings of *IEEE Congress on Evolutionary Computation 2011*, pp: 1057-1062.

S. Bandyopadhyay and A. Mukherjee. 2015. An algorithm for many-objective optimization with reduced objective computations: a study in differential evolution. *IEEE Transactions on Evolutionary Computation,* vol. 19(3), pp: 400-413.

A. Basak, D. Maity, and S. Das. 2013. A differential invasive weed optimization algorithm for improved global numerical optimization. *Applied Mathematics and Computation*, vol. 219(12), pp: 6645-6668.

A. Basak, S. Das, and K. C. Tan. 2013. Multimodal optimization using a biobjective differential evolution algorithm enhanced with mean distance-based selection. *IEEE Transactions on Evolutionary Computation*, vol. 17(5), pp: 666-685.

B. Biswal, H. S. Behera, R. Bisoi, and P. K. Dash. 2012. Classification of power quality data using decision tree and chemotactic differential evolution based fuzzy clustering. *Swarm and Evolutionary Computation*, vol. 4, pp: 12–24.

S. Biswas, S. Kundu, S. Das and A. V. Vasilakos. 2013. Teaching and learning best differential evolution with self adaptation for real parameter optimization. In proceedings of *IEEE Congress on Evolutionary Computation* 2013, June 20-23, Cancún, México, pp: 1115-1122.

S. Biswas, S. Das, P. N. Suganthan, and C. A. C. Coello. 2014a. Evolutionary multiobjective optimization in dynamic environments: A set of novel benchmark functions. In proceedings of *IEEE Congress on Evolutionary Computation*, pp: 3192 - 3199.

S. Biswas, S. Kundu, and S. Das. 2014b. An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution. *IEEE Transactions on Cybernetics*, vol. 44(10), pp: 1726-1737.

S. Biswas, S. Kundu, and S. Das. 2015. Including niching behavior in differential evolution through local information sharing. *IEEE Transactions on Evolutionary Computations*, Vol. 19(2): 246-263, 2015.

C. Blum, J. Puchinger, G. R. Raidl, and A. Roli. 2011. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing,* vol. 11(6), pp: 4135 – 4151.

I. Boussaïd, A. Chatterjee, P. Siarry, and M. Ahmed-Nacer, 2011. Hybridizing biogeography-based optimization with differential evolution for optimal power allocation in wireless sensor networks. *IEEE T. Vehicular Technology*, vol. 60(5), pp: 2347-2353.

J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer. 2006. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, vol. 10(6) no. 6, pp. 646–657.

J. Brest, A. Zamuda, B. Bošković, M. S. Maučec, and V. Žumer. 2008. High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction. In proceedings of 2008 *IEEE World Congress on Computational Intelligence*, IEEE Press, pp: 2032–2039.

J. Brest, A. Zamuda, I. Fister, and M. S. Maučec. 2010. Large scale global optimization using self-adaptive differential evolution algorithm. In proceedings of *IEEE Congress on Evolutionary Computation*, Barcelona, Spain, pp: 1-8.

J. Brest and M. Maučec. 2011. Self-adaptive differential evolution algorithm using population size reduction and three strategies. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 15, pp. 2157–2174.

J. Brest, B. Bošković, A. Zamuda, I. Fister, and E. Mezura-Montes. 2013. Real parameter single objective optimization using self-adaptive differential evolution algorithm with more strategies. In proceedings of *IEEE Congress on Evolutionary Computation 2013*, June 20-23, Cancún, México, pp: 377-383.

J. Brest, A. Zamuda, B. Bošković. 2015. *Adaptation and Hybridization in Computational Intelligence*, Springer International Publishing.

P. Bujok, J. Tvrdík, and R. Poláková. 2014. Differential evolution with rotation-invariant mutation and competing-strategies adaptation, in proceedings of *IEEE Congress on Evolutionary Computation 2014*, July 6-11, Beijing, China, pp: 2253-2258.

P. Bujok and J. Tvrdík. 2012. Parallel migration model employing various adaptive variants of differential evolution. In Lecture Notes in Computer Science, vol. 7269, pp: 39-47.

S. Bureerat. 2011. Improved population-based incremental learning in continuous spaces. *Advances in Intelligent and Soft Computing*, vol. 96, pp: 77–86.

Z. Cai, W. Gong, C. X. Ling, and H. Zhang. 2011. A clustering-based differential evolution for global optimization. *Applied Soft Computing*, vol. 11, pp: 1363-1379.

Y. Cai and J. Wang. 2013. Differential evolution with neighborhood and direction information for numerical optimization. *IEEE Transactions on Cybernetics*, vol. 43(6), pp: 2202-2215.

T. Chakraborti, A. Chatterjee, A. Halder, and A. Konar. 2015. Automated emotion recognition employing a novel modified binary quantum-behaved gravitational search algorithm with differential mutation. *Expert Systems*, vol. 32(4), pp: 522-530.

L. Chang, C. Liao, W. Lin, L. L. Chen, and X. Zheng, 2012. A hybrid method based on differential evolution and continuous ant colony optimization and its application on wideband antenna design. *Progress in Electromagnetics Research*, vol. 122, pp: 105-118.

Q. Chen, B. Liu, Q. Zhang, J. J. Liang, P. N. Suganthan, B. Y. Qu. 2014. Problem Definition and Evaluation Criteria for CEC 2015 Special Session and Competition on Bound Constrained Single-Objective Computationally Expensive Numerical Optimization. *Technical Report, Computational Intelligence Laboratory Zhengzhou University, China and Nanyang Technological University*, Singapore, Nov 2014.

Y. Chen, W. Xie, and X. Zou. 2015. A binary differential evolution algorithm learning from explored solutions. *Neurocomputing*, vol. 149, pp: 1038-1047.

Z. Chen, X. Jiang, J. Li, S. Li and L. Wang. 2013. PDECO: parallel differential evolution for cluster optimization. *Journal of Computational Chemistry*, vol. 34(12), pp: 1046-1059.

C. W. Chiang, W. P. Lee, and J. S. Heh. 2010. A 2-Opt based differential evolution for global optimization. *Applied Soft Computing*, vol. 10, pp. 1200-1207.

G. A. Croes. 1958. A method for solving traveling—salesman problems, *Operations Research*, vol. 6(6), pp: 791–812.

S. Das, A. Konar, U. K. Chakraborty, 2005. Two improved differential evolution schemes for faster global search, in: *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, 2005, pp. 991–998.

S. Das, A. Abraham, U. K. Chakraboty, and A. Konar. 2009. Differential evolution using a neighborhood-based mutation operator. *IEEE Transactions on Evolutionary Computation*, vol. 13(3), pp: 526–553.

S. Das, A. Ghosh, and S. S. Mullick. 2015. A switched parameter differential evolution for large scale global optimization – simpler may be better, MENDEL 2015: 21st *International Conference on Soft Computing, Advances in Intelligent Systems and Computing* Vol. 378, pp 103 -125, Brno, Czech Republic, June 23 – 25, 2015.

S. Das, A. Mandal, and R. Mukherjee. 2014. An adaptive differential evolution algorithm for global optimization in dynamic environments. *IEEE Transactions on Cybernetics*, vol. 44(6), pp: 966-978.

S. Das and P. N. Suganthan. 2011. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation,* vol. 15(1), pp:4-31.

R. Dash, P. K. Dash and R Bisoi. 2014. A self-adaptive differential harmony search based optimized extreme learning machine for financial time series prediction. *Swarm and Evolutionary Computation*, vol. 19, pp: 25-42.

K. Deb. 2000. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering,* vol. 186, pp: 311–338.

K. Deb. 2001. *Multi-objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, UK.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, vol. 6(2), Pp: 182–197.

G. Deng and X. Gu. 2012. A hybrid discrete differential evolution algorithm for the no-idle permutation flow shop scheduling problem with makespan criterion. *Computers & OR*, vol. 39(9), pp. 2152-2160.

R. Denysiuk, L. Costa, and I. E. Santo. 2013. Many-objective optimization using differential evolution with variable-wise mutation restriction. In proceedings of GECCO'13, pp: 591-598.

B. Dorronsoro and P. Bouvry. 2011. Improving classical and decentralized differential evolution with new mutation operator and population topologies. *IEEE Transactions on Evolutionary Computation*, vol. 15(1), pp: 67-98.

A. Draa, S. Bouzoubia, and I. Boukhalfa. 2015. A sinusoidal differential evolution algorithm for numerical optimization. *Applied Soft Computing*, vol. 27, pp: 99-126.

E.-N. Dragoi and V. Dafinescu, Parameter control and hybridization techniques in differential evolution: a survey, *Artificial Intelligence Review*, DOI 10.1007/s10462-015-9452-8, 2015.

D. Datta and J. R. Figueira. 2013. A real–integer–discrete-coded differential evolution. *Applied Soft Computing*, Vol. 13, pp: 3884-3893.

M. A. Eita and A. A. Shoukry. 2014. Constrained dynamic differential evolution using a novel hybrid constraint handling technique. In proceedings of *IEEE International Conference on Systems, Man, and Cybernetics*, San Diego, CA, USA.

S. M. Elsayed, R. A. Sarker and D. L. Essam. 2011. Differential evolution with multiple strategies for solving CEC2011 real-world numerical optimization problems. In proceedings of *IEEE Congress on Evolutionary Computation, 2011*, pp: 1041-1048.

S. M. Elsayed, R. A. Sarker and D. L. Essam. 2011b. GA with a new multi-parent crossover for solving IEEE-CEC2011 competition problems," In proceedings of *IEEE Congress on Evolutionary Computation, 2011*, pp: 1034-1040.

S. M. Elsayed, R. A. Sarker and T. Ray. 2013. Differential evolution with automatic parameter configuration for solving the CEC2013 competition on real-parameter optimization. In proceedings of *IEEE Congress on Evolutionary Computation, 2013*, June 20-23, Cancún, México, pp: 1932-1937.

S. M. Elsayed, T. Ray and R. A. Sarker. 2015. A surrogate-assisted differential evolution algorithm with dynamic parameters selection for solving expensive optimization problems. In proceedings of *IEEE Congress on Evolutionary Computation, 2014,* July 6-11, Beijing, China, pp: 1062-1068.

M. G. Epitropakis, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis. 2011a. Enhancing differential evolution utilizing proximity-based mutation operators. *IEEE Transactions on Evolutionary Computation*, vol. 15(1), pp: 99-118.

M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis. 2011b. Finding multiple global optima exploiting differential evolution's niching capability. In proceedings of *2011 IEEE Symposium on Differential Evolution* (SDE), April 2011, pp. 1-8.

M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis. 2012. Evolving cognitive and social experience in particle swarm optimization through differential evolution: A hybrid approach. *Information Sciences*, vol. 216, pp: 50-92.

M. G. Epitropakis, Li, X., and E. K. Burke. 2013. A dynamic archive niching differential evolution algorithm for multimodal optimization. In proceedings of *IEEE Congress on Evolutionary Computation*, *2013*. CEC 2013. Cancun, Mexico, pp. 79-86.

I. De Falco, A. Della Cioppa, D. Maisto, U. Scafuri, and E. Tarantino. 2007. Satellite image registration by distributed differential evolution. In *Applications of Evolutionary Computing, Lectures Notes in Computer Science*, vol.4448, Springer, pp: 251–260.

I. D. Falco, A. D. Cioppa, D. Maisto, U. Scafuri, and E. Tarantino. 2014. An adaptive invasion-based model for distributed differential evolution. *Information Sciences*, vol. 278, pp: 53-672.

W. Gao, G. G. Yen, and S. Liu. 2014. A cluster-based differential evolution with self-adaptive strategy for multimodal optimization. *IEEE Transactions on Cybernetics*, vol. 44(8), pp: 1314-1327.

X. Z. Gao, X. Wang, S. J. Ovaska, and K. Zenger. 2014. A hybrid optimization method based on differential evolution and harmony search. *International Journal of Computational Intelligence and Applications*, Vol. 13(1).

W. F. Gao, G. G. Yen, and S. Y. Liu. 2015. A dual-population differential evolution with coevolution for constrained optimization, *IEEE Transactions on Cybernetics*, early access.

C. García-Martínez, F. J. Rodríguez, and M. Lozano. 2011. Role differentiation and malleable mating for differential evolution: an analysis on large-scale optimization. *Soft Computing*, vol. 15, pp: 2109–2126.

A. Ghosh, S. Das, A. Chowdhury, and R. Giri. 2011. An improved differential evolution algorithm with fitness-based adaptation of the control parameters. *Information Sciences*, vol. 181, pp: 3749-3765.

S. Ghosh, S. Das, S. Roy, S. K. M. Islam, and P. N. Suganthan. 2012a. A differential covariance matrix adaptation evolutionary algorithm for real parameter optimization. *Information Sciences*, vol. 182, pp: 199-219.

S. Ghosh, S. Das, A. V. Vasilakos, and K. Suresh. 2012b. On convergence of differential evolution over a class of continuous functions with unique global optimum. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics,* vol. 42(1), pp: 107-124.

A. Glotić and A. Zamuda. 2015. Short-term combined economic and emission hydrothermal optimization by surrogate differential evolution. *Applied Energy*, Vol. 141, pp. 42-56.

D. E. Goldberg and J. Richardson. 1987. Genetic algorithms with sharing for multimodal function optimization. In proceedings of *2nd International Conference on Genetic Algorithms*, pp: 41–49.

D. E. Goldberg. 1990. Probability matching, the magnitude of reinforcement, and classifier system bidding. *Mach. Learn*., vol. 5 (4), pp: 407–425.

W. Gong, Z. Cai, C. X. Ling, and H. Li. 2011a. Enhanced differential evolution with adaptive strategies for numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 41(2), pp: 397-413.

W. Gong, A. Fialho, Z. Cai, and H. Li. 2011b. Adaptive strategy selection in differential evolution for numerical optimization: An empirical study. *Information Sciences*, vol. 181, pp: 5364-5386.

W. Gong and Z. Cai. 2013. Differential evolution with ranking-based mutation operators. *IEEE Transactions on Cybernetics*, vol. 43(6), pp: 2066-2081.

W. Gong, Z. Cai, and Y. Wang. 2014. Repairing the crossover rate in adaptive differential evolution. Applied Soft Computing, vol.: 15, pp: 149-168.

W. Gong, Z. Cai, and D. Liang. 2015. Adaptive ranking mutation operator based differential evolution for constrained optimization. *IEEE Transactions on Cybernetics*, early access.

V. Gonuguntla, R. Mallipeddi, and Kalyana C. Veluvolu. 2015. Differential Evolution with population and strategy parameter adaptation, *Mathematical Problems in Engineering*, Vol. 2015, Article ID 287607.

H. Guo, Y. Li, J. Li, H. Sun, D. Wang, and X. Chen. 2014. Differential evolution improved with self-adaptive control parameters based on simulated annealing, *Swarm and Evolutionary Computation*, vol. 19, pp 52-67, December 2014.

S. M. Guo, C. C. Yang, P. H. Hsu, and J. S. H. Tsai. 2015. Improving differential evolution with successful parent selecting framework, *IEEE Transactions on Evolutionary Computation*, early access.

S. M. Guo and C. C. Yang. 2015. Enhancing differential evolution utilizing eigenvector-based crossover operator, *IEEE Transactions on Evolutionary Computation*, early access.

U. Halder, S. Das, and D. Maity. 2013. A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments, *IEEE Transactions on Cybernetics*, vol. 43(3), pp: 881-897.

Y. He, X. Wang, K. Liu, and Y. Wang, (2010). Convergent analysis and algorithmic improvement of differential evolution. *Journal of Software*, 21(5):875–885.

Z. B. Hu, S. W. Xiong, Q. H. Su, and Z. X. Fang. 2014. Finite Markov chain analysis of classical differential evolution algorithm. *Journal of Computational and Applied Mathematics*, vol. 268, pp: 121-134.

S. Hui and P. N. Suganthan. 2014. Niching-based Self-adaptive Ensemble DE with MMTS for solving Dynamic Optimization Problems. *Proc. of 2014 IEEE Congress on Evolutionary Computation*, Beijing, July.

S. Hui and P. N. Suganthan. 2015. Ensemble and arithmetic recombination-based speciation differential evolution for multimodal optimization. *IEEE Transactions on Cybernetics*, Early Access.

G. Iacca, R. Mallipeddi, E. Mininno, F. Neri, and P. N. Suganthan. Super-fit and population size reduction in compact differential evolution. In *IEEE SSCI 2011 - Symposium Series on Computational Intelligence* - MC 2011: 2011 *IEEE Workshop on Memetic Computing*, pages 21–28, Paris, France, 2011.

S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan. 2012. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *IEEE Transactions on Systems, Man, And Cybernetics—Part B: Cybernetics*, vol. 42(2), pp: 482-500.

D. Jia, G. Zheng, and M. K. Khan. 2011. An effective memetic differential evolution algorithm based on chaotic local search. *Information Sciences*, vol. 181, pp: 3175-3187.

G. Jia, Y. Wang, Z. Cai, and Y. Jin. 2013. An improved (λ+μ) constrained differential evolution for constrained optimization. *Information Sciences*, vol. 222, pp: 302-322.

S. Jiang and S. Yang. 2015. An improved multi-objective optimization evolutionary algorithm based on decomposition for complex Pareto fronts. *IEEE Transactions on Cybernetics*, early access, DOI: 10.1109/TCYB.2015.2403131.

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

J. Kennedy. 2003. Bare bones swarms. *Proc. of IEEE Swarm Intell. Symp.*, pp: 80–87.

S. Kitayama, M. Arakawa and K. Yamazaki. 2011. Differential evolution as the global optimization technique and its application to structural optimization. *Applied Soft Computing*, vol. 11, pp: 3792-3803.

P. Krömer, J. Platoš, V. Snášel, and A. Abraham. 2013, Many-threaded differential evolution on the GPU, *Massively Parallel Evolutionary Computation on GPGPUs*, Part of the series Natural Computing Series pp 121-147

S. Kundu, S. Biswas, S. Das, and P. N. Suganthan. 2013. Crowding-based local differential evolution with speciation-based memory archive for dynamic multimodal optimization, in proceedings of *GECCO'13*, July 6–10, 2013, Amsterdam, The Netherlands, pp: 33-40.

W. B. Langdon and R. Poli 2007. Evolving problems to learn about particle swarm optimizers and other search algorithms, *IEEE Trans. Evol. Comput*., vol. 11, no. 5, pp. 561–578.

Y. W. Leung, and Y. Wang. 2001. An orthogonal genetic algorithm with quantization for global numerical optimization, *IEEE Transactions on Evolutionary Computation,* vol. 5(1), pp: 41–53.

Y. Li, and J. Zhang. 2011. A new differential evolution algorithm with dynamic population partition and local restart, in proceedings of *GECCO'11*, July 12–16, Dublin, Ireland, Pp: 1085-1092.

Z. li, Z. Shang, B.Y. Qu, J. J. Liang. 2014. Differential evolution strategy based on the constraint of fitness values classification. In proceedings of *IEEE Congress on Evolutionary Computation 2014*, July 6-11, Beijing, China, pp: 1454-1460.

Y. Li, Z. Zhan, Y. Gong, W. Chen, J. Zhang, and Y. Li. 2015. Differential evolution with an evolution path: a DEEP evolutionary algorithm. *IEEE Transactions on Cybernetics*, early access.

J. J. Liang, P. N. Suganthan and K. Deb, "Novel composition test functions for numerical global optimization," IEEE Swarm Intelligence Symposium, pp. 68-75, Pasadena, CA, USA, June 2005.

J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernández-Díaz. 2013a. Problem definitions and evaluation criteria for the CEC 2013 special session and competition on real-parameter optimization, *Technical Report 201212*, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore.

J. J. Liang, B. Y. Qu, and P. N. Suganthan. 2013b. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization, *Technical Report 201311*, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore.

J. J. Liang, B. Y. Qu, X. B. Mao, B. Niu, and D. Y. Wang. 2014. Differential evolution based on fitness Euclidean distance ratio for multi modal optimization. *Neurocomputing*, vol. 137, pp: 152-260.

J. J. Liang, B. Y. Qu, P. N. Suganthan, Q. Chen. 2014b. Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-based Real-Parameter Single Objective Optimization", *Technical Report, Computational Intelligence Laboratory, Zhengzhou University, China and Nanyang Technological University, Singapore*, Nov.

T. W. Liao. 2010. Two hybrid differential evolution algorithms for engineering design optimization. *Applied Soft Computing*, vol. 10, pp: 1188-1199.

H. Liu, Z. Cai, and Y. Wang. 2010. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing*, vol. 10, pp: 629-640.

G. Liu, Y. Li, X. Niea, and H. Zheng. 2012. A novel clustering-based differential evolution with 2 multi-parent crossovers for global optimization. *Applied Soft Computing*, vol. 12, pp: 663-681.

E. D. López, A. Puris, and R. R. Bello. 2015. VMODE: A Hybrid Metaheuristic for the Solution of Large Scale Optimization Problems. *Revista Investigacion Operacional*, vol. 36(3), pp: 232-239.

M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and Mauro Birattari. The irace package, Iterated Race for Automatic Algorithm Configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université libre de Bruxelles, Belgium, 2011.

M. Lozano, F. Herrera, N. Krasnogor, and D. Molina. 2004. Real-coded memetic algorithms with crossover hill-climbing. *Evolutionary Computation Journal*, vol. 12(3), pp: 273–302.

X. Lu, K. Tang, B. Sendhoff, and X. Yao. 2014. A new self-adaptation scheme for differential evolution, *Neurocomputing*, vol. 146, pp: 2-16.

X.-F. Lu and K. Tang, 2012. Classification and regression assisted differential evolution for computationally expensive problems, Journal of Computer Science and Technology vol. 27(5) pp: 1024-1034.

N Lynn, P N Suganthan. 2015. Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation. *Swarm and Evolutionary Computation* 24, pp: 11-24.

F. D. Maio, S. Baronchelli, and E. Zio. 2015. Hierarchical differential evolution for minimal cut sets identification: Application to nuclear safety systems. *European Journal of Operational Research*, vol. 242, pp. 10–20.

R. Mallipeddi, 2013. Harmony search based parameter ensemble adaptation for differential evolution, *Journal of Applied Mathematics*, Volume 2013, Article ID 750819.

R. Mallipeddi, G. Wu, M. Lee, P. N. Suganthan, 2014. Gaussian adaptation based parameter adaptation for differential evolution. *IEEE Congress on Evolutionary Computation*, Beijing, China, pp. 1760-1767.

R. Mallipeddi and P. N. Suganthan. 2010. Ensemble of constraint handling techniques, *IEEE Transactions on Evolutionary Computation*, Volume 14, Issue 4, pages 561-579.

R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren. 2011. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, vol. 11, pp: 1670-1696.

R Mallipeddi and M Lee, An evolving surrogate model-based differential evolution algorithm. 2015. Applied Soft Computing 34, 770-787.

A. L. Maravilha, J. A. Ramírez, and F. Campelo. 2014. Combinatorial optimization with differential evolution: a set-based approach. In proceedings of the *2014 Conference on Genetic and Evolutionary Computation (GECCO) Companion*, ACM New York, NY, USA, pp. 69-70.

V. V. D. Melo, and A. C. B. Delbem. 2012. Investigating smart sampling as a population initialization method for differential evolution in continuous problems. *Information Sciences*, vol. 193, pp: 36-53.

R. Mendes, 2004. *Population Topologies and Their Influence in Particle Swarm Performance*. Dissertation, University of Minho.

R. Mendes and A. Mohais. 2005. DynDE: A differential evolution for dynamic optimization problems. In proceedings of *IEEE Congress on Evolutionary Computation 2005*, pp: 2808–2815.

E. Mezura-Montes, M. E. Miranda-Varela, and R. C. Gómez-Ramón. 2010. Differential evolution in constrained numerical optimization: An empirical study. *Information Sciences*, vol. 180, pp: 4223-4262.

E. Mininno, F. Neri, F. Cupertino, and D. Naso. 2011. Compact differential evolution. *IEEE Transactions on Evolutionary Computation*, vol. 15(1), pp: 32-54.

V. Miranda and R. Alves. 2013. Differential Evolutionary Particle Swarm Optimization (DEEPSO): a successful hybrid. In proceedings of *BRICS Congress on Computational Intelligence & 11th Brazilian Congress on Computational Intelligence*, pp. 368 - 374.

S. J. A. Miruna and S. Baskar. 2015. Surrogate assisted-hybrid differential evolution algorithm using diversity control. *Expert Systems*, vol. 32, pp: 531-545.

A. W. Mohamed and H. Z. Sabry. 2012. Constrained optimization based on modified differential evolution algorithm. *Information Sciences*, vol. 194, pp: 171-208.

A. W. Mohamed, An improved differential evolution algorithm with triangular mutation for global numerical optimization, Computers & Industrial Engineering, Vol. 85 (2015) 359–375.

R. Mukherjee, G. R. Patra, R. Kundu, and S. Das. 2014. Cluster-based differential evolution with crowding archive for niching in dynamic environments. *Information Sciences*, vol. 267, pp: 58-82.

J. A. Nelder, and R. Mead. 1965. A simplex-method for function minimization. *Computer Journal*, vol. 7(4), pp: 308–313.

F. Neri, G. Iacca, and E. Mininno. 2011. Disturbed exploitation compact differential evolution for limited memory optimization problems. *Information Sciences*, vol. 181, pp: 2469-2487.

F. Neri and V. Tirronen. 2010. Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review*, vol. 33(1-2), pp: 61-106.

T. T. Nguyen, S. Yang, and J. Branke. 2012. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24.

J. Olenšěk, T. Tuma, J. Puhan, and A. Buŕmen. 2011. A new asynchronous parallel global optimization method based on simulated annealing and differential evolution. *Applied Soft Computing*, vol. 11, pp: 1481-1489.

M. N. Omidvar, X. Li, Y. Mei, and X. Yao. 2014. Cooperative Co-Evolution with Differential Grouping for Large Scale Optimization. *IEEE Transactions on Evolutionary Computation*, vol. 18(3), pp: 378-393.

Y. S. Ong and A. J. Keane. 2004. Meta-Lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Comput*ation, vol. 8(2), pp: 99–110.

K. Pal, C. Saha, S. Das. 2013. Differential evolution and offspring repair method based dynamic constrained optimization. *Swarm, Evolutionary, and Memetic Computing*, vol. 8297 of the series Lecture Notes in Computer Science, pp 298-309.

D. R. Penas, J. R. Banga, P. González and R. Doallo. 2014. A parallel differential evolution algorithm for parameter estimation in dynamic models of biological systems. In Advances in Intelligent Systems and Computing, vol. 294, pp: 173-181.

N. Pholdee and S. Bureerat. 2013. Hybridization of real-code population-based incremental learning and differential evolution for multiobjective design of trusses. *Information Sciences*, vol. 223, pp: 136-152.

A. P. Piotrowski. 2013. Adaptive memetic differential evolution with global and local neighborhood-based mutation operators. *Information Sciences*, vol. 241, pp: 164-194, 2013.

M. C. D. Plessis, and A. P. Engelbrecht. 2012. Using competitive population evaluation in a differential evolution algorithm for dynamic environments. *European Journal of Operational Research*, vol: 218, pp: 7-20.

I. Poikolainen, and F. Neri. 2013. Differential evolution with concurrent fitness based local search, in proceedings of *IEEE Congress on Evolutionary Computation 2013*, June 20-23, Cancún, México, pp: 384-391.

I. Poikolainen, F. Neri, and F. Caraffini. 2015. Cluster-based population initialization for differential evolution frameworks. *Information Sciences*, vol. 297, pp: 216-235.

R. Poláková, J. Tvrdík, and P. Bujok. 2014. Controlled restart in differential evolution applied to CEC2014 benchmark functions, in proceedings of *IEEE Congress on Evolutionary Computation 2014*, July 6-11, Beijing, China, pp: 2230-2236.

D. Powell and M. M. Skolnick. 1993. Using genetic algorithms in engineering design optimization with non-linear constraints. *Proceedings of the 5th international conference on genetic algorithms*, 17–21 July, Urbana-Champaign, IL, USA. ACM Press, 424–431.

K. V. Price, R. Stron and J. Lampinen. 2005. *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Germany, Springer.

A. Puris, R. Bello, D. Molina, and F. Herrera. 2012. Variable mesh optimization for continuous optimization problems. *Soft Computing*, vol. 16(3), pp: 511-525.

A. K. Qin and P. N. Suganthan. 2005. Self-adaptive Differential Evolution Algorithm for Numerical Optimization", *IEEE Congress on Evolutionary Computation*, pp: 1785-1791, Edinburgh, UK, Sept.

A. K. Qin, V. L. Huang, and P. N. Suganthan. 2009. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, vol. 13(2), pp: 398–417.

B. Y. Qu and P. N. Suganthan. 2011. Multi-objective differential evolution based on the summation of normalized objectives and improved selection method. *SDE-2011, IEEE Symposium on Differential Evolution*, pp. 1-8, Paris, France, DOI: 10.1109/SDE.2011.5952065, April.

B. Y. Qu, P. N. Suganthan, and J. J. Liang. 2012. Differential evolution with neighborhood mutation for multimodal optimization. *IEEE Transactions on Evolutionary Computation*, vol: 16(5), pp: 601-614.

B. Y. Qu, P. N. Suganthan and S. Das. 2013. A distance-based locally informed particle swarm model for multimodal optimization. *IEEE Transactions on Evolutionary Computation,* vol. 17(3), pp: 387-402.

B. Y. Qu, J. J. Liang, J. M. Xiao, and Z. G. Shang. 2014. Memetic differential evolution based on fitness Euclidean-distance ratio. In proceedings of *IEEE Congress on Evolutionary Computation* 2014, July 6-11, Beijing, China, pp: 2266-2273.

B. Y. Qu, J. J. Liang, Z. Y. Wang, Q. Chen, P. N. Suganthan. 2015. Novel benchmark functions for continuous multimodal optimization with comparative results, *Swarm and Evolutionary Computation*, doi:10.1016/j.swevo.2015.07.003.

S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama. 2008. Opposition-based differential evolution. *IEEE Transactions on Evolutionary Computation*, vol. 12(1), pp: 64-79.

P. Rakshit, A. Konar, P. Bhowmik, I. Goswami, S. Das, L. C. Jain, and A. K. Nagar. 2013. Realization of an adaptive memetic algorithm using differential evolution and q-learning: a case study in multirobot path planning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43(4), pp: 814-831.

P. Rakshit, A. Konar, S. Das, L. C. Jain, and A. K. Nagar. 2014. Uncertainty management in differential evolution induced multiobjective optimization in presence of measurement noise. *IEEE Transactions On Systems, Man, and Cybernetics*: *Systems*, vol. 44(7), pp: 922-937.

G. Reynoso-Meza, J. Sanchis, X. Blasco and J. M. Herrero. 2011. Hybrid DE algorithm with adaptive crossover operator for solving real-world numerical optimization problems. In proceedings of *IEEE Congress on Evolutionary Computation,* 2011, pp: 1551-1556.

T. Robič and Filipič. 2015. DEMO: Differential evolution for multiobjective optimization. In proceedings of *3rd International Conference Evolutionary Multi-Criterion Optimization*, LNCS 3410, 2005, pp: 520–533.

H. H. Rosenbrock. 1960. An automatic method for finding the greatest or least value of a function, *Comput. J.,* vol. 3 (3), pp. 175–184.

P. J. Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, *J. Comput. Appl. Math*., vol. 20, pp: 53–65.

J. L. Rueda, I. Erlich. 2015. MVMO for bound constrained single-objective computationally expensive numerical optimization. Proc. IEEE CEC, pp: 1011-1017.

C. Saha, S. Das, K. Pal, and S. Mukherjee. 2015. Fuzzy rule-based penalty function approach for constrained optimization. *IEEE Transactions on Cybenetics*, early access, DOI: 10.1109/TCYB.2014.2359985.

S. Sardar, S. Maity, S. Das, and P. N. Suganthan. 2011. Constrained real parameter optimization with a gradient repair based Differential Evolution algorithm. In proceedings of *SDE,* 2011, pp:1-8.

R. A. Sarker, S. M. Elsayed, and T. Ray. 2014. Differential evolution with dynamic parameters selection for optimization problems. *IEEE Transactions on Evolutionary Computation*, vol. 18(5), pp: 689-707.

E. Sayed, D. Essam, R. Sarker, S. Elsayed. 2015. Decomposition-based evolutionary algorithm for large scale constrained problems. *Information Sciences*, vol. 316, pp: 457–486.

R. C. P. Silva, R. A. Lopes, and F. G. Guimarães. 2011. Self-adaptive mutation in the differential evolution. In proceedings of *GECCO'11*, July 12–16, 2011, Dublin, Ireland, pp: 1939-1946.

K. Sindhya, S. Ruuska, T. Haanp, and K. Miettinen. 2011. A new hybrid mutation operator for multi objective optimization with differential evolution. *Soft Computing*, vol. 15(10), pp: 2041–2055.

S. K. Smit, A. E. Eiben 2010. Beating the 'world champion' evolutionary algorithm via REVAC Tuning. *IEEE Congress on Evolutionary Computation* (CEC), 2010, pp. 1-8.

W.M. Spears, "Adapting crossover in evolutionary algorithms", in: J.R. McDonnell, R.G. Reynolds, D.B. Fogel (Eds.), The 4th *Annual Conference on Evolutionary Programming*, MIT Press, 1995, pp. 367–384.

R. Storn and K. V. Price, "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," ICSI, USA, Tech. Rep. TR-95-012, 1995 [Online]. Available: http://icsi.berkeley.edu/~storn/litera.html.

R. Storn and K. Price, "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

P. N. Suganthan. 2015. Numerical optimization by nature inspired algorithms. Keynote Speeches at BRIC CCI 2015, ICSI 2015, ICHSA 2015. Available from: http://www.ntu.edu.sg/home/epnsugan/

P. N. Suganthan. 1999. Particle swarm optimisation with a neighbourhood operator. *Proc. Congress on Evolutionary Computation*, Washington DC, USA, July.

C. F. Sun, Differential evolution and its application on the optimal scheduling of electrical power system, Ph.D. thesis, Huazhong University of Science and Technology, 2009.

A. M. Sutton, M. Lunacek, and L. D. Whitley. 2007. Differential evolution and non-separability: using selective pressure to focus search, in proceeding of *GECCO'07,* pp: 1428-1435.

R. Tanabe and A. Fukunaga. 2013. Success-history based parameter adaptation for differential evolution. In proceedings of *IEEE Congress on Evolutionary Computation* 2013, June 20-23, Cancún, México, pp: 71–78.

R. Tanabe and A. S. Fukunaga. 2014. Improving the search performance of shade using linear population size reduction. In proceedings of *IEEE Congress on Evolutionary Computation 2014*, July 6-11, Beijing, China, pp: 1658-1665.

R. Tanabe and A. Fukunaga. 2014b. Reevaluating exponential crossover in differential evolution. *Proc. Parallel Problem Solving from Nature* Ljubljana, September, pp: 201-210.

L. Tang, Y. Dong, and J. Liu. 2015. Differential evolution with an individual-dependent mechanism. *IEEE Transactions on Evolutionary Computation*, early access.

M. F. Tasgetiren, Ö. Bulut, M. M. Fadiloglu, 2011. A differential evolution algorithm for the economic lot scheduling problem. *IEEE Symposium on Differential Evolution* (SDE) 2011: 164-169, Paris, France.

M. F. Tasgetiren, Q. K. Pan, P. N. Suganthan, and O. Buyukdagli. 2013. A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem. *Computers & OR*, vol. 40(7), pp: 1729-1743.

M. F. Tasgetiren, Q.-K. Pan, D. Kizilay, G. A. Süer, 2015. A differential evolution algorithm with variable neighborhood search for multidimensional knapsack problem*. IEEE Congress on Evolutionary Computation* (CEC) 2015, pp. 2797-2804, Japan.

D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis. 2004. Parallel differential evolution. In proceedings of *IEEE Congress on Evolutionary Computation*, 2004, pp: 2023–2029.

D. Thierens. 2005. An adaptive pursuit strategy for allocating operator probabilities. In proceedings of *Genetic Evol. Comput. Conf. 2005*, pp: 1539–1546.

R. Thomsen. 2004. Multimodal optimization using Crowding-based differential evolution. In proceedings of *Congress on Evolutionary Computation*, 19-23 June 2004, pp: 1382-1389.

D. H. Tran, M. Y. Cheng, M. T. Cao. 2015. Hybrid multiple objective artificial bee colony with differential evolution for the time–cost–quality tradeoff problem. *Knowledge-Based Systems*, vol. 74, pp: 176–186.

A. Trivedi, D. Srinivasan, S. Biswas, and T. Reindl. 2015. Hybridizing genetic algorithm with differential evolution for solving the unit commitment scheduling problem. *Swarm and Evolutionary Computation*, vol. 23, pp: 50-64.

J. Tvrdík, "Self-adaptive variants of differential evolution with exponential crossover", *Analele of West University Timisoara, Series Mathematics-Informatics*, vol. 47, pp. 151–168, 2009.

J. Tvrdík, R. Poláková, J. Veselský, and P. Bujok. 2012. Adaptive variants of differential evolution: Towards control-parameter-free optimizers, in *Handbook of Optimization*. Springer, pp: 423–449.

S. M. Venske, R. A. Gonçalves, and M. R. Delgado. 2014. ADEMO/D: Multi objective optimization by an adaptive differential evolution algorithm. *Neurocomputing*, vol. 127, pp: 65-77.

S. Wan and D. Wang. 2013. A novel differential evolution for dynamic multiobjective optimization with adaptive immigration scheme. In proceedings of 3rd *International Conference on Computer Science and Network Technology* (ICCSNT), Oct. 2013, Dalian, China, pp. 502 – 507.

H. Wang, Z. Wu, S. Rahnamayan and D. Jiang. 2010. Sequential DE enhanced by neighborhood search for large scale global optimization. In proceedings of *IEEE Congress on Evolutionary Computation*, Barcelona, Spain, pp: 1-7.

Y. Wang and Z. Cai. 2011a. Constrained evolutionary optimization by means of (λ+μ)-differential evolution and improved adaptive trade-off model. *Evolutionary Computation*, vol. 19(2), pp: 249–285.

Y. Wang, Z. Cai, and Q. Zhang. 2011b. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, vol. 15(1), pp: 55-66.

L. Wang and L. Li. 2011. Fixed-structure H_∞ controller synthesis based on differential evolution with level comparison, *IEEE Transactions on Evolutionary Computation,* vol. 15(1), pp. 120-129.

L. Wang, X. Fu, Y. Mao, M. I. Menhas, and M. Fei. 2012. A novel modified binary differential evolution algorithm and its applications. *Neurocomputing*, vol. 98, pp. 55-75.

H. Wang, S. Rahnamayan, H. Sun, and M. G. H. Omran. 2013. Gaussian bare-bones differential evolution. *IEEE Transactions on Cybernetics*, vol. 43(2), pp: 634-647.

H. Wang, Z. J. Wu, and S. Rahnamayan. 2011. Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. *Soft Computing*, vol. 15(11), pp: 2127–2140.

H. Wang, S. Rahnamayan, Z. Wu. 2013. Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems. *Journal of Parallel and Distributed Computing*, vol. 73, pp: 62-73.

J. Wang, J. Liao, Y. Zhou, and Y. Cai. 2014. Differential evolution enhanced with multiobjective sorting-based mutation operators, *IEEE Transactions on Cybernetics*, vol. 44(12), pp: 2792-2805.

Y. Wang, and Z. Cai. 2012. Combining multi-objective optimization with differential evolution to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, vol. 16(1), pp: 117-134.

Y. Wang, Z. Cai, and Q. Zhang. 2012. Enhancing the search ability of differential evolution through orthogonal crossover. *Information Sciences*, vol. 185, pp: 153-177.

Y. Wang, H. Li, T. Huang, and L. Li. 2014. Differential evolution based on covariance matrix learning and bimodal distribution parameter setting. *Applied Soft Computing*, vol. 18, pp: 232-247.

M. Weber, F. Neri, V. Tirronen. 2011a. Shuffle or update parallel differential evolution for large-scale Optimization. *Soft Computing*, vol. 15, pp: 2089–2107.

M. Weber, F. Neri, and V. Tirronen. 2011b. A study on scale factor in distributed differential evolution. *Information Sciences*, vol. 181, pp: 2488-2511.

T. H. Wong, A. K. Qin, S. Wang, and Y. Shi, 2014. cuSaDE: A CUDA-Based Parallel Self-adaptive Differential Evolution Algorithm, Proceedings of the 18th *Asia Pacific Symposium on Intelligent and Evolutionary Systems* - Volume 2 of the series Proceedings in Adaptation, Learning and Optimization, pp 375-388.

G.H. Wu, R. Mallipeddi, P. N. Suganthan, R. Wang, and H. Chen. 2015a. differential evolution with multi population based ensemble of mutation strategies, *Information Sciences*, DoI: 10.1016/j.ins.2015.09.009.

G.H. Wu, W. Pedrycz, P. N. Suganthan, R. Mallipeddi. 2015b. Variable reduction strategy for evolutionary algorithms handling equality constraints. *Applied Soft Computing*, DoI: 10.1016/j.asoc.2015.09.007.

B. Xin, J. Chen, J. Zhang, H. Fang, and Z. Peng. 2012. Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and taxonomy. *IEEE Transactions on Systems, Man, and Cybernetics*, *Part C*, vol. 42(5), pp: 744-767.

C. Xu, H. Huang, and S. Ye. 2014. A differential evolution with replacement strategy for real-parameter numerical optimization. In proceedings of *IEEE Congress on Evolutionary Computation*, July 6-11, 2014, Beijing, China, pp: 1617-1624

Z. Yang, K. Tang, and X. Yao. 2008. Self-adaptive differential evolution with neighborhood search. In Proceedings of *IEEE Congress on Evolutionary Computation*, June 2008, pp: 1110–1116.

M. Yang, Z. Cai, C. Li, and J. Guan. 2013. An improved adaptive differential evolution algorithm with population adaptation. In *Proceedings of the 15th annual conference on Genetic and Evolutionary Computation* (*GECCO* '13), Christian Blum (Ed.). ACM, New York, NY, USA, 145-152.

M. Yang, C. Li, Z. Cai, and J. Guan. 2015. Differential evolution with auto-enhanced population diversity. *IEEE Transactions on Cybernetics*, vol. 45(2).

W. Yu and J. Zhang. 2011. Multi-population differential evolution with adaptive parameter control for global optimization. In proceedings of *GECCO'11*, July 12–16, 2011, Dublin, Ireland, pp: 1093-1098.

W. Yu, M. Shen, W. Chen, Z. Zhan, Y. Gong, Y. Lin, O. Liu, and J. Zhang. 2014a. Differential evolution with two-level parameter adaptation. *IEEE Transactions on Cybernetics*, vol. 44(7), Pp: 1080-1099.

W. J. Yu, J. J. Li, J. Zhang, and M. Wan. 2014b. Differential evolution using mutation strategy with adaptive greediness degree control. In proceedings of *GECCO '14*, July 12–16, 2014, Vancouver, BC, Canada, pp: 73-79.

A. Zamuda and J. Brest. 2012. Population reduction differential evolution with multiple mutation strategies in real world industry challenges. In *Swarm and Evolutionary Computation - International Symposia*, SIDE 2012 and EC 2012, pages 154–161, Zakopane, Poland, 2012.

A. Zamuda, J. Brest, and E. Mezura-Montes. 2013. Structured population size reduction differential evolution with multiple mutation strategies on CEC 2013 real parameter optimization. *IEEE Congress on Evolutionary Computation* (CEC) 2013, 2013, pp. 1925-1931.

A. Zamuda and J Brest. 2015. Self-adaptive control parameters: randomization frequency and propagations in differential evolution. *Swarm and Evolutionary Computation*, vol. 25, pp: 72-99.

E. Zhabitskaya and M. Zhabitsky. 2013a. Asynchronous differential evolution with adaptive correlation matrix. In proceeding of *GECCO'13*, July 6–10, 2013, Amsterdam, The Netherlands, pp: 455-462.

E. Zhabitskaya and M. Zhabitsky. 2013b. Asynchronous differential evolution with restart. *Lecture Notes in Computer Science*, vol. 8236, pp: 555-561, 2013.

Z. H. Zhan and J. Zhang. 2010. Self-adaptive differential evolution based on PSO learning strategy. In proceedings of *GECCO'10*, pp: 39-47.

Z. Zhan and J. Zhang. 2012. Enhanced differential evolution with random walk. In proceedings of *GECCO'12 Companion*, July 7–11, 2012, Philadelphia, PA, USA, pp: 1513-1514.

J. Zhang and A. C. Sanderson. 2009a. JADE: Adaptive differential evolution with optional external archive, *IEEE Transactions on Evolutionary Computation*, vol. 13(5), pp. 945–958.

J. Zhang and A. C. Sanderson. 2009b. Adaptive differential evolution: a robust approach to multimodal problem optimization, Berlin, Germany: Springer-Verlag.

C. Zhang, J. Chen, and B. Xin. 2013. Distributed memetic differential evolution with the synergy of Lamarckian and Baldwinian learning. *Applied Soft Computing*, vol. 13, pp: 2947-2959.

Q. Zhang and H. Li, MOEA/D: A multi-objective evolutionary algorithm based on decomposition, *IEEE Trans. on Evolutionary Computation*, vol.11, no. 6, pp712-731 2007.

Q. Zhang, W. Liu, and H Li, The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances, Working Report CES-491, School of CS & EE, University of Essex, 02/2009.

S. Z. Zhao and P. N. Suganthan, S. Das. 2011. Self-adaptive differential evolution with multi-trajectory search for large scale optimization. *Soft Computing,* November, Vol 15, Issue 11, pp: 2175-2185.

S. Z Zhao, P N Suganthan, and Q Zhang, 2012. MOEA/D with an ensemble of neighbourhood sizes, *IEEE Trans on Evolutionary Computation*, vol. 16(3), pp: 442-446.

S. Z. Zhao and P. N. Suganthan. 2013. Empirical investigations into the exponential crossover of differential evolutions. *Swarm and Evolutionary Computation*, vol. 9 pp: 27-36.

Y. Zhao, J. Wang, and Y. Song, (2009). An improved differential evolution to continuous domains and its convergence. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 1061–1064. ACM.

Y. J. Zheng, X. L. Xu, H. F. Ling, and S. Y. Chen. 2015. A hybrid fireworks optimization method with differential evolution operators. *Neurocomputing*, vol. 148, pp: 75–82, Jan., 2015.

J. H. Zhong and J. Zhang. 2011. Adaptive multi-objective differential evolution with stochastic coding strategy, in proceedings of *GECCO*'11, July 12–16, 2011, Dublin, Ireland, pp: 665-672.

J. Zhong and J. Zhang. 2012. SDE: A stochastic coding differential evolution for global optimization. In proceedings of *GECCO'12*, July 7-11, 2012, Philadelphia, Pennsylvania, USA, pp: 975-981.

J. Zhong, M. Shen, J. Zhang, H. S. Chung, Y. Shi, and Yun Li.2013. A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem. *IEEE Transactions on Evolutionary Computation*, vol. 17(4), pp: 512-527.

Y. Zhou, X. Li, and L. Gao, 2013. A differential evolution algorithm with intersect mutation operator. *Applied Soft Computing,* vol. 13, pp. 390-401.

W. Zhu, Y. Tang, J. Fang, and W. Zhang. 2013. Adaptive population tuning scheme for differential evolution. *Information Sciences*, vol. 223, pp. 164-191.

D. Zou, J. Wu, L. Gao, and S. Li. 2013. A modified differential evolution algorithm for unconstrained optimization problems. *Neurocomputing,* vol. 120, pp: 469-481.

Supplementary Document

(DE Application References corresponding to Tables 3 - 5)

A. Al-Ani, A. Alsukker, and R. N. Khushaba. 2013. Feature subset selection using differential evolution and a wheel based search strategy. *Swarm and Evolutionary Computation*, vol. 9, pp:15–26.

M. Ali, C. W. Ahn, and M. Pant. 2014. Multi-level image thresholding by synergetic differential evolution. *Applied Soft Computing*, vol. 17, pp: 1-11.

R. A. Aliev, W. Pedrycz, B. G. Guirimov, R. R. Aliev, U. Ilhan, M. Babagil, and S. Mammadli. 2011. Type-2 fuzzy neural networks with fuzzy clustering and differential evolution optimization. *Information Sciences*, vol. 181, pp: 1591-1608.

S. M. Almeida-Luz, M. A. Vega-Rodríguez, J A. Gómez-Púlido. 2011. and J. M. Sánchez-Pérez, Differential evolution for solving the mobile location management. *Applied Soft Computing* , vol. 11, pp: 410-427.

N. Baatar, K. Jeong, and C. Koh. 2014. Adaptive parameter controlling non-dominated ranking differential evolution for multi-objective optimization of electromagnetic problems. *IEEE Transactions on Magnetics*, vol. 50(2), pp: 709-712.

A. Banerjee, and I. Abu-Mahfouz. 2014. A comparative analysis of particle swarm optimization and differential evolution algorithms for parameter estimation in nonlinear dynamic systems. *Chaos, Solitons & Fractals: Nonlinear Science, and Nonequilibrium and Complex Phenomena*, vol. 58, pp: 65-83.

M. Basu. 2011. Economic environmental dispatch using multi-objective differential evolution. *Applied Soft Computing*, vol. 11, pp: 2845-2853.

Y. Bazi, N. Alajlan, F. Melgani, H. A. Hichri, S. Malek, and R. R. Yager. 2014. Differential evolution extreme learning machine for the classification of hyperspectral images. *IEEE Geoscience and Remote Sensing Letters*, vol. 11(6), pp: 1066-1070.

Tapas Bhadra, and Sanghamitra Bandyopadhyay. 2015. Unsupervised Feature Selection using an Improved version of Differential Evolution, *Expert Systems with Applications*, vol. 42(8), pp: 4042–4053.

A. Bhattacharya, and P. K. Chattopadhyay. 2011. Solving economic emission load dispatch problems using hybrid differential evolution. *Applied Soft Computing*, vol. 11, pp: 2526-2537.

A. K. Bhattacharya, D. Aditya, D. Sambasivam. 2013. Estimation of operating blast furnace reactor invisible interior surface using differential evolution, *Applied Soft Computing,* vol. 13, pp: 2767-2789.

W. D. Chang. 2012. Differential evolution-based nonlinear system modeling using a bilinear series model, *Applied Soft Computing*, vol. 12, pp: 3401-3407.

C. Chen, and S. Yang. 2014. Neural fuzzy inference systems with knowledge-based cultural differential evolution for nonlinear system control. *Information Sciences,* vol. 270, pp: 154-171.

Y. Chen, V. Mahalec, Y. Chen, X. Liu, R. He, and K. Sun. 2015. Reconfiguration of satellite orbit for cooperative observation using variable-size multi-objective differential evolution. *European Journal of Operational Research*, vol. 242, pp: 10-20.

L. D. S. Coelho , V. C. Mariani , M. V. F. D. Luz , and J. V. Leite. 2013. Novel gamma differential evolution approach for multiobjective transformer design optimization. *IEEE Transactions on Magnetics*, vol. 49(5), pp: 2121-2124.

A. Deb, J. S. Roy, and B. Gupta. 2014. Performance comparison of differential evolution, particle swarm optimization and genetic algorithm in the design of circularly polarized microstrip antennas. *IEEE Transactions on Antennas and Propagation*, vol. 62(8), pp: 3920-3928.

H. Dhahri, A. M. Alimi, and A. Abraham. 2012. Hierarchical multi-dimensional differential evolution for the design of beta basis function neural network. *Neurocomputing*, vol. 97, pp: 131-140.

C. Dong, W. W. Y. Ng, X. Wang, P. P. K. Chan, and D. S. Yeung. 2014. An improved differential evolution and its application to determining feature weights in similarity-based clustering. *Neurocomputing*, vol. 146, pp: 95-103.

E. Dragoi, S. Curteanu, A. Galaction, and D. Cascaval. 2013. Optimization methodology based on neural networks and self-adaptive differential evolution algorithm applied to an aerobic fermentation process. *Applied Soft Computing*, vol. 13, pp: 222-238.

I. D. Falco. 2013. Differential evolution for automatic rule extraction from medical databases. *Applied Soft Computing*, vol. 13, pp: 1265-1283.

Y. Fu, M. Ding, C. Zhou, and H. Hu. 2013. Route planning for unmanned aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43(6).

Z. Gao, Z. Pan, and J. Gao. 2014. A new highly efficient differential evolution scheme and its application to waveform inversion. *IEEE Geoscience and Remote Sensing Letters*, vol. 11(10), pp: 1702-1706.

M. Ghasemi, M. M. Ghanbarian, S. Ghavidel, S. Rahmani, and E. M. Moghaddam. 2014. Modified teaching learning algorithm and double differential evolution algorithm for optimal reactive power dispatch problem: A comparative study. *Information Sciences*, vol. 278, pp: 231-249.

A. Ghosh, A. Datta, S. Ghosh. 2013. Self-adaptive differential evolution for feature selection in hyper-spectral image data. *Applied Soft Computing*, vol. 13, pp: 1969-1977.

A. Ghosh, A. Mondal, and S. Ghosh. 2014. Moving object detection using markov random field and distributed differential evolution. *Applied Soft Computing*, vol. 15, pp: 121-135.

P. Ghosh, S. Das, and H. Zafar. 2012. Adaptive differential-evolution-based design of two-channel quadrature mirror filter banks for sub-band coding and data transmission. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 42(6): 1613-1623.

A. Glotić, A. Glotić, P. Kitak, J. Pihler, and I. Tičar. 2014. Parallel self-adaptive differential evolution algorithm for solving short-term hydro scheduling problem. *IEEE Transactions on Power Systems*, vol. 29(5), pp: 2347-2358.

A. Glotić and A. Zamuda. 2015. Short-term combined economic and emission hydrothermal optimization by surrogate differential evolution. *Applied Energy*, Vol. 141, pp. 42-56.

S. Gundry, J. Zou, M. U. Uyar, C. S. Sahin, and J. Kusyk. 2015. Differential evolution-based autonomous and disruption tolerant vehicular self-organization in MANETs. *Ad Hoc Networks*, vol. 25, pp: 454-471.

N. Hachicha, B. Jarboui, and P. Siarry. 2011. A fuzzy logic control using a differential evolution algorithm aimed at modeling the financial market dynamics. *Information Sciences*, vol. 181, pp: 79-91.

M. M. Joly, T. Verstraete, G. Paniagua. 2013. Differential evolution based soft optimization to attenuate vane–rotor shock interaction in high-pressure turbines. *Applied Soft Computing*, vol. 13, pp: 1882-1891.

L. Kang, L. Wu, X. Chen, and Y. Yang. 2013. Practical structure and motion recovery from two uncalibrated images using ε constrained adaptive differential evolution. *Pattern Recognition*, vol. 46, pp: 1466-1484.

D. Karaboga, and S. Okdem. 2004. A simple and global optimization algorithm for engineering problems: differential evolution algorithm. *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 12 (1), pp: 53–60.

R. N. Khushaba, A. Al-Ani, A. Al-Jumaily. 2011. Feature subset selection using differential evolution and a statistical repair mechanism. *Expert Systems with Applications*, vol. 38, pp: 11515–11526.

D. Kranjcic and G. Štumberger. 2014. Differential evolution-based identification of the nonlinear Kaplan turbine model. *IEEE Transactions on Energy Conversion*, vol. 29(1), pp: 178-187.

P. Kuila, and P. K. Jana. 2014. A novel differential evolution based clustering algorithm for wireless sensor networks. *Applied Soft Computing*, vol. 25, pp: 414-425.

S. Kumar, D. Datta, and B. V. Babu. 2011. Estimation of equilibrium parameters using differential evolution in reactive extraction of propionic acid by tri-n-butyl phosphate. *Chemical Engineering and Processing*, vol. 50, pp: 614– 622.

W. Kwedlo. 2011. A clustering method combining differential evolution with the K-means algorithm, *Pattern Recognition Letters*, vol.32, pp: 1613-1621.

J. C. Y. Lai, F. H. F. Leung, S. H. Ling, and H. T. Nguyen. 2013. Hypoglycaemia detection using fuzzy inference system with multi-objective double wavelet mutation differential evolution. *Applied Soft Computing*, vol. 13, pp: 2803-2811.

B. Lei, I. Y. Soon, and E. Tan. 2013. Robust SVD-based audio watermarking scheme with differential evolution optimization, *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21(11), pp: 2368-2378.

F. Lezama, G. Castañón, and A. M. Sarmiento. 2012. Differential evolution optimization applied to the wavelength converters placement problem in all optical networks. *Computer Networks*, vol. 56, pp: 2262-2275.

X. Li, and M. Yin. 2012. Application of differential evolution algorithm on self-potential data. *PLOS ONE*, vol. 7(12), number: e51199.

Y. F. Li, G. Sansavini, and E. Zio. 2013. Non-dominated sorting binary differential evolution for the multi-objective optimization of cascading failures protection in complex networks. *Reliability Engineering and System Safety*, vol. 111, pp: 195-205.

T. W. Liao, P. J. Egbelu, and P.C. Chang. 2012. Two hybrid differential evolution algorithms for optimal inbound and outbound truck sequencing in cross docking operations. *Applied Soft Computing*, vol. 12, pp: 3683-3697.

W. Liu, P. Wang, and H. Qiao. 2012. Part-based adaptive detection of work pieces using differential evolution. *Signal Processing*, vol. 92, pp: 301-307.

N. B. Lovett, C. Crosnier, M. Perarnau-Llobet, and B. C. Sanders. 2013. Differential evolution for many-particle adaptive quantum metrology. *Physical Review Letters,* vol. 110(22), pp: 220501-1 - 220501-5.

H. Lu, M. Chang, and C. Tsai. 2012. Parameter estimation of fuzzy neural network controller based on a modified differential evolution. *Neurocomputing*, vol. 89, pp: 178-192.

P. Luukka, J. Lampinen. 2011. Differential evolution classifier in noisy settings and with interacting variables. *Applied Soft Computing*, vol. 11, pp: 891-899.

F. D. Maio, S. Baronchelli, and E. Zio. 2014. Hierarchical differential evolution for minimal cut sets identification: Application to nuclear safety systems. *European Journal of Operational Research*, vol. 238, pp: 645-652.

R Mallipeddi, J. P. Lie, S. G. Razul, P. N. Suganthan. 2011 CMS See, Robust adaptive beamforming based on covariance matrix reconstruction for look direction mismatch, *PIER Letters*, 25, 37-46.

R Mallipeddi, JP Lie, PN Suganthan, SG Razul, CMS See. 2011 A differential evolution approach for robust adaptive beamforming based on joint estimation of look direction and array geometry, PIER Research 119, 381-394.

R Mallipeddi, J. P. Lie, P. N. Suganthan, S. G Razul, CMS See. 2011 Near optimal robust adaptive beamforming approach based on evolutionary algorithm, Progress In Electromagnetics Research B 29, 157-174.

R. Mallipeddi, S. Jeyadevi, P. N. Suganthan, and S. Baskar. 2012. Efficient constraint handling for optimal reactive power dispatch problems, *Swarm and Evolutionary Computation*, Vol. 5, Pages 28–36.

P. Mesejo, R. Ugolotti, F. D. Cunto, M. Giacobini, and S. Cagnoni. 2013. Automatic hippocampus localization in histological images using Differential Evolution-based deformable models. *Pattern Recognition Letters*, vol. 34, pp: 299-307.

F. Neri, and V. Tirronen. Scale factor local search in differential evolution. *Memetic Computing*, vol. 1(2), pp: 153–171.

J. Novo, J. Santos, and M. G. Penedo. 2013. Multiobjective differential evolution in the optimization of topological active models, *Applied Soft Computing*, vol. 13, pp: 3167-3177.

C. Nyirarugira and T. Y. Kim. 2013. Adaptive differential evolution algorithm for real time object tracking. *IEEE Transactions on Consumer Electronics*, vol. 59(4), pp: 833-838.

S. Oh, W. Kim, W. Pedrycz, and S. Joo. 2012. Design of K-means clustering-based polynomial radial basis function neural networks (pRBF NNs) realized with the aid of particle swarm optimization and differential evolution. *Neurocomputing*, vol. 78, pp: 121-132.

Q. Pan, L. Wang, L. Gao, and W. D. Li. 2011. An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers. *Information Sciences*, vol. 181, pp: 668-685.

M. Pandit, L. Srivastava, and M. Sharma. 2015. Environmental economic dispatch in multi-area power system employing improved differential evolution with fuzzy selection. *Applied Soft Computing*, vol. 28, pp: 498-510.

A. K. Parwani, P. Talukdar, and P. M. V. Subbarao. 2013. Performance evaluation of hybrid differential evolution approach for estimation of the strength of a heat source in a radiatively participating medium. *International Journal of Heat and Mass Transfer*, vol. 56, pp: 552-560.

S. Paul and S. Das. 205. Simultaneous feature selection and weighting - an evolutionary multi-objective optimization approach, Pattern Recognition Letters, Vol. 65, Pages 51–59.

A. P. Piotrowski. 2014. Differential evolution algorithms applied to neural network training suffer from stagnation. *Applied Soft Computing*, Vol. 21, Pp: 382-406.

A. Ponsich, and C. A. C. Coello. 2013. A hybrid differential evolution—tabu Search algorithm for the solution of job-shop scheduling problems. *Applied Soft Computing*, vol. 13, pp: 462-474.

F. Pop, D. Pallez, M. Cremene, A. G. B. Tettamanzi, M. Suciu, and M. Vaida. 2011. QoS-Based Service Optimization using Differential Evolution, in proceedings of *GECCO'11*, July 12–16, 2011, Dublin, Ireland, pp:1891-1898.

R. S. Prado, R. C. P. Silva, O. M. Neto, F. G. Guimarães, D. S. Sanches, J. B. A. London Jr., and A. C. B. Delbem. 2014. Differential evolution using ancestor tree for service restoration in power distribution systems, *Applied Soft Computing*, vol. 23, pp: 498-508.

K. V. Price, R. M. Storn, and J. A. Lampinen. 2005. Differential evolution – a practical approach to global optimization, Springer, Berlin.

S. N. Qasem, and S. M. Shamsuddin. 2011. Memetic elitist pareto differential evolution algorithm based radial basis function networks for classification problems. *Applied Soft Computing*, vol. 11, pp: 5565-5581.

G. Quaranta, G. C. Marano, R. Greco, and G. Monti. 2014. Parametric identification of seismic isolators using differential evolution and particle swarm optimization. *Applied Soft Computing*, vol. 22, pp: 458-464.

T. Raghunathan, and D. Ghose. 2014. Differential evolution based 3-D guidance law for a realistic interceptor model, *Applied Soft Computing*, vol. 16, pp: 20-23.

P. Rocca, G. Oliveri, and A. Massa. 2011. Differential Evolution as Applied to Electromagnetics. *IEEE Antennas and Propagation Magazine*, vol. 53(1), pp:38-49.

G. Sannino, I. D. Falco, and G. D. Pietro. 2014. Monitoring obstructive sleep apnea by means of a real-time mobile system based on the automatic extraction of sets of rules through differential evolution. *Journal of Biomedical Informatics*, vol. 49, pp: 84-100.

S. Sarkar, S. Das, and S. S. Chaudhuri. 2015. A multilevel color image thresholding scheme based on minimum cross entropy and differential evolution. *Pattern Recognition Letters*, vol. 54, pp: 27-35.

H. Saruhan. 2014. Differential evolution and simulated annealing algorithms for mechanical systems design, *Engineering Science and Technology, an International Journal*, vol. 17, pp: 131-136.

S. Sayah, and A. Hamouda. 2013. A hybrid differential evolution algorithm based on particle swarm optimization for non-convex economic dispatch problems. *Applied Soft Computing*, vol. 13, pp: 1608-1619.

O. Schleusing, T. Kinnunen, B. Story, and J. Vesin. 2013. Joint source-filter optimization for accurate vocal tract estimation using differential evolution. *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21(8), pp: 1560-1572.

S. Sharma, and G. P. Rangaiah. 2013. An improved multi-objective differential evolution with a termination criterion for optimizing chemical processes. *Computers and Chemical Engineering*, vol. 56, pp: 155– 173.

M. Secmen, M. F. Tasgetiren, and K. Karabulut, 2013. Null control in linear antenna arrays with ensemble differential evolution. *IEEE Symposium on Differential Evolution* (SDE) 2013, pp. 92-98, Singapore.

S. Sengupta, S. Das, Md. Nasir, A. V. Vasilakos, and W. Pedrycz. 2012. An evolutionary multiobjective sleep-scheduling scheme for differentiated coverage in wireless sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 42(6): 1093-1102.

D. O. Silva, L. G. M. Vieira, F. S. Lobato, and M. A. S. Barrozo. 2012. Optimization of the design and performance of hydrocyclones by differential evolution technique. *Chemical Engineering and Processing*, vol. 61, pp: 1– 7.

S. Sivasubramani, and K. S. Swarup. 2012. Multiagent based differential evolution approach to optimal power flow, *Applied Soft Computing*, vol. 12, pp: 736-740.

B. Subudhi, and D. Jena. 2011a. Nonlinear system identification using memetic differential evolution trained neural networks. *Neurocomputing*, vol. 74, pp: 1696-1709.

B. Subudhi, D. Jena. 2011b. A differential evolution based neural network approach to nonlinear system identification. *Applied Soft Computing*, vol. 11, pp: 861-871.

Y. Tang, X. Zhang, C. Hua, L. Li, and Y Yang. 2012. Parameter identification of commensurate fractional-order chaotic system via differential evolution. *Physics Letters A*, vol. 376, pp: 457-464.

L. Tang, Y. Zhao, and J. Liu. 2014. An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production. *IEEE Transactions on Evolutionary Computation*, vol. 18(2), pp: 267-281.

I. Triguero, S. García, and F. Herrera. 2011. Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification. *Pattern Recognition*, vol. 44, pp: 901-916.

I. Triguero, J. Derrac, S. García, and F. Herrera. 2012. Integrating a differential evolution feature weighting scheme into prototype generation. *Neurocomputing*, vol. 97, pp: 332-343.

J. T. Tsai, W. H. Ho, J. H. Chou, and C. Y. Guo. 2011. Optimal approximation of linear systems using Taguchi-sliding-based differential evolution algorithm. *Applied Soft Computing*, vol. 11, pp: 2007-2016.

J. Tsai. 2015. Improved differential evolution algorithm for nonlinear programming and engineering design problems. *Neurocomputing*, vol. 148, pp: 628-640.

R. Ugolotti, Y. S. G. Nashed, P. Mesejo, Š. Ivekovič, L. Mussi, and S. Cagnoni. 2013a. Particle swarm optimization and differential evolution for model-based object detection. *Applied Soft Computing* , vol. 13, pp: 3092-3105.

R. Ugolotti, and S. Cagnoni. 2013b. Differential evolution based human body pose estimation from point clouds, in proceedings of *GECCO'13*, July 6–10, 2013, Amsterdam, The Netherlands, pp: 1389-1396.

P. Upadhyay, R. Kar, D. Mandal, and S. P. Ghoshal. 2014. IIR system identification using differential evolution with wavelet mutation. *Engineering Science and Technology, an International Journal,* vol.17, pp: 8-24.

R. Vakili, P. Setoodeh, E. Pourazadi, D. Iranshahi, and M. R. Rahimpour. 2011. Utilizing differential evolution (DE) technique to optimize operating conditions of an integrated thermally coupled direct DME synthesis reactor. *Chemical Engineering Journal*, vol. 168, pp: 321–332.

V. S. Vakula, and K. R. Sudha. 2012. Design of differential evolution algorithm-based robust fuzzy logic power system stabilizer using minimum rule base. *IET Generation, Transmission & Distribution*, vol. 6(2), pp: 121–132.

M. Vasile, E. Minisci, and M. Locatelli. 2011. An inflationary differential evolution algorithm for space trajectory optimization. *IEEE Transactions on Evolutionary Computation*, vol. 15(2), pp: 209-225.

L. W. H. Vincent and S. G. Ponnambalam. 2013. A differential evolution-based algorithm to schedule flexible assembly lines. *IEEE Transactions on Automation Science and Engineering*, vol. 10(4), pp: 1161-1165.

A. R. Yildiz. 2013a. A new hybrid differential evolution algorithm for the selection of optimal machining parameters in milling operations. *Applied Soft Computing*, vol. 13, pp: 1561-1566.

A. R. Yildiz. 2013b. Hybrid Taguchi-differential evolution algorithm for optimization of multi-pass turning operations, *Applied Soft Computing*, vol. 13, pp: 1433-1439.

A. Zamuda, J. Brest, B. Bŏsković, and V. Žumer. 2011. Differential evolution for parameterized procedural woody plant models reconstruction. *Applied Soft Computing*, vol. 11, pp: 4904-4912.

A. Zamuda, and J. D. H. Sosa. 2014a. Differential evolution and underwater glider path planning applied to the short-term opportunistic sampling of dynamic meso-scale ocean structures. *Applied Soft Computing*, vol. 24, pp: 95-108.

A. Zamuda, and J. Brest. 2014b. Vectorized procedural models for animated trees reconstruction using differential evolution. *Information Sciences*, vol. 278, pp: 1-21.

S. Zhai, and T. Jiang. 2015. A new sense-through-foliage target recognition method based on hybrid differential evolution and self-adaptive particle swarm optimization-based support vector machine. *Neurocomputing*, vol. 149, pp: 573-584.

C. Zhan, W. Situ, L. F. Yeung, P. W. Tsang, and G. Yang. 2014. A parameter estimation method for biological systems modeled by ODE/DDE models using spline approximation and differential evolution algorithm. *IEEE/ACM Transactions On Computational Biology And Bioinformatics*, vol. 11(6), pp: 1066-1076.

G. Zhang, J. Cheng, M. Gheorghe, and Q. Meng. 2013. A hybrid approach based on differential evolution and tissue membrane systems for solving constrained manufacturing parameter optimization problems. *Applied Soft Computing*, vol. 13, pp: 1528-1542.

R. Zhang, S. Song, and C. Wu. 2013. A hybrid differential evolution algorithm for job shop scheduling problems with expected total tardiness criterion. *Applied Soft Computing*, vol. 13, pp: 1448-1458.

X. Zhang, and H. Duan. 2014. An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning. *Applied Soft Computing*, vol. 16, pp: 270-284.

Y. Zhao, F. Chen, Q. Shen, and L. Zhang. 2012. Optimizing low loss silver nanowires structure metamaterial at yellow light spectrum with differential evolution. *Physics Letters A*, vol. 376, pp: 352-356.

Y.-x. Zhao, W. Li, S. Feng, W. Y. Ochieng, and W. Schuster. 2014. An Improved Differential Evolution Algorithm for Maritime Collision Avoidance Route Planning, *Abstract and Applied Analysis*, vol. 2014, Article ID 614569, 10 pages, doi:10.1155/2014/614569.

J. Zhao, Y. Xu, F. Luo, Z. Y. Dong, and Y. Peng. 2014. Power system fault diagnosis based on history driven differential evolution and stochastic time domain simulation. *Information Sciences*, vol. 275, pp: 13-29.

Y. Zhong, and L. Zhang. 2012. Remote sensing image subpixel mapping based on adaptive differential evolution. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 42(5), pp: 1306-1329,

Y. Zhong, S. Zhang, and L. Zhang. 2013. Automatic fuzzy clustering based on adaptive multi-objective differential evolution for remote sensing imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6(5), pp: 2290-2301.

W. Zhu, J. Fang, Y. Tang, W. Zhang, and W. Du. 2012. Digital IIR filters design using differential evolution algorithm with a controllable probabilistic population size. *PLOS ONE*, vol. 7(7), number: e40549.

W. Zhu, J. Fang, Y. Tang, W. Zhang, and Y. Xu. 2012. Identification of fractional-order systems via a switching differential evolution subject to noise perturbations. *Physics Letters A*, vol. 376, pp: 3113-3120.

E. Zio, and G. Viadana. 2011. Optimization of the inspection intervals of a safety system in a nuclear power plant by Multi-Objective Differential Evolution (MODE). *Reliability Engineering and System Safety*, vol. 96, pp: 1552-1563.

E. Zio, L. R. Golea, and G. Sansavini. 2012. Optimizing protections against cascades in network systems: A modified binary differential evolution algorithm. *Reliability Engineering and System Safety*, vol. 103, pp: 72-83.