# Differential Evolution: A Survey of the State-of-the-Art

Swagatam Das, *Member, IEEE,* and Ponnuthurai Nagaratnam Suganthan, *Senior Member, IEEE*

***Abstract*—Differential evolution (DE) is arguably one of the most powerful stochastic real-parameter optimization algorithms in current use. DE operates through similar computational steps as employed by a standard evolutionary algorithm (EA). However, unlike traditional EAs, the DE-variants perturb the current-generation population members with the scaled differences of randomly selected and distinct population members. Therefore, no separate probability distribution has to be used for generating the offspring. Since its inception in 1995, DE has drawn the attention of many researchers all over the world resulting in a lot of variants of the basic algorithm with improved performance. This paper presents a detailed review of the basic concepts of DE and a survey of its major variants, its application to multiobjective, constrained, large scale, and uncertain optimization problems, and the theoretical studies conducted on DE so far. Also, it provides an overview of the significant engineering applications that have benefited from the powerful nature of DE.**

***Index Terms*—Derivative-free optimization, differential evolution (DE), direct search, evolutionary algorithms (EAs), genetic algorithms (GAs), metaheuristics, particle swarm optimization (PSO).**

## I. Introduction

TO TACKLE complex computational problems, researchers have been looking into nature for years—both as model and as metaphor—for inspiration. Optimization is at the heart of many natural processes like Darwinian evolution itself. Through millions of years, every species had to adapt their physical structures to fit to the environments they were in. A keen observation of the underlying relation between optimization and biological evolution led to the development of an important paradigm of computational intelligence—the evolutionary computing techniques [S1]–[S4] for performing very complex search and optimization.

Evolutionary computation uses iterative progress, such as growth or development in a population. This population is then selected in a guided random search using parallel processing to achieve the desired end. The paradigm of evolutionary computing techniques dates back to early 1950s, when the idea to use Darwinian principles for automated problem solving originated. It was not until the sixties that three distinct interpretations of this idea started to be developed in three different places. Evolutionary programming (EP) was introduced by Lawrence J. Fogel in the USA [S5],[1] while almost simultaneously. I. Rechenberg and H.-P. Schwefel introduced evolution strategies (ESs) [S6], [S7] in Germany. Almost a decade later, John Henry Holland from University of Michigan at Ann Arbor, devised an independent method of simulating the Darwinian evolution to solve practical optimization problems and called it the genetic algorithm (GA) [S8]. These areas developed separately for about 15 years. From the early 1990s on they are unified as different representatives ("dialects") of one technology, called evolutionary computing. Also since the early nineties, a fourth stream following the same general ideas started to emerge—genetic programming (GP) [S9].

Nowadays, the field of nature-inspired metaheuristics is mostly constituted by the evolutionary algorithms [comprising of GAs, EP, ESs, GP, differential evolution (DE), and so on] as well as the swarm intelligence algorithms [e.g., ant colony optimization (ACO), particle swarm optimization (PSO), Bees algorithm, bacterial foraging optimization (BFO), and so on [S10]–[S12]]. Also the field extends in a broader sense to include self-organizing systems [S13], artificial life (digital organism) [S14], memetic and cultural algorithms [S15], harmony search [S16], artificial immune systems [S17], and learnable evolution model [S18].

The DE [72], [73], [88]–[90] algorithm emerged as a very competitive form of evolutionary computing more than a decade ago. The first written article on DE appeared as a technical report by R. Storn and K. V. Price in 1995 [88]. One year later, the success of DE was demonstrated at the First International Contest on Evolutionary Optimization in May 1996, which was held in conjunction with the 1996 IEEE International Conference on Evolutionary Computation (CEC) [89]. DE finished third at the First International Contest on Evolutionary Optimization (1st ICEO), which was held in Nagoya, Japan. DE turned out to be the best evolutionary algorithm for solving the real-valued test function suite of the 1st ICEO (the first two places were given to non-evolutionary algorithms, which are not universally applicable but solved the test-problems faster than DE). Price presented DE at the Second International Contest on Evolutionary Optimization in

Manuscript received September 17, 2009; revised March 9, 2010 and June 10, 2010; accepted June 12, 2010. Date of publication October 14, 2010; date of current version February 25, 2011. This work was supported by the Agency for Science, Technology, and Research, Singapore (A∗Star), under Grant #052 101 0020.

S. Das is with the Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata 700 032, India (e-mail: swagatamdas19@yahoo.co.in).

P. N. Suganthan is with the School of Electrical and Electronic Engineering, Nanyang Technological University, 639798, Singapore (e-mail: epnsugan@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2010.2059031

[1]Due to space limitation the supplementary reference list cited as [Sxxx] will not be published in print, but as an on-line document only.

1997 [72] and it turned out as one of the best among the competing algorithms. Two journal articles [73], [92] describing the algorithm in sufficient details followed immediately in quick succession. In 2005 CEC competition on real parameter optimization, on 10-D problems classical DE secured 2nd rank and a self-adaptive DE variant called SaDE [S201] secured third rank although they performed poorly over 30-D problems. Although a powerful variant of ES, known as restart covariance matrix adaptation ES (CMA-ES) [S232, S233], yielded better results than classical and self-adaptive DE, later on many improved DE variants [like improved SaDE [76], jDE [10], opposition-based DE (ODE) [82], DE with global and local neighborhoods (DEGL) [21], JADE [118], and so on that will be discussed in subsequent sections] were proposed in the period 2006–2009. Hence, another rigorous comparison is needed to determine how well these variants might compete against the restart CMA-ES and many other real parameter optimizers over the standard numerical benchmarks. It is also interesting to note that the variants of DE continued to secure front ranks in the subsequent CEC competitions [S202] like CEC 2006 competition on constrained real parameter optimization (first rank), CEC 2007 competition on multiobjective optimization (second rank), CEC 2008 competition on large scale global optimization (third rank), CEC 2009 competition on multiobjective optimization (first rank was taken by a DE-based algorithm MOEA/D for unconstrained problems), and CEC 2009 competition on evolutionary computation in dynamic and uncertain environments (first rank). We can also observe that no other single search paradigm such as PSO was able to secure competitive rankings in all CEC competitions. A detailed discussion on these DE-variants for optimization in complex environments will be provided in Section V.

In DE community, the individual trial solutions (which constitute a population) are called *parameter vectors* or *genomes*. DE operates through the same computational steps as employed by a standard EA. However, unlike traditional EAs, DE employs difference of the parameter vectors to explore the objective function landscape. In this respect, it owes a lot to its two ancestors namely—the Nelder-Mead algorithm [S19], and the controlled random search (CRS) algorithm [S20], which also relied heavily on the difference vectors to perturb the current trial solutions. Since late 1990s, DE started to find several significant applications to the optimization problems arising from diverse domains of science and engineering. Below, we point out some of the reasons why the researchers have been looking at DE as an attractive optimization tool and as we shall proceed through this survey, these reasons will become more obvious.

1) Compared to most other EAs, DE is much more simple and straightforward to implement. Main body of the algorithm takes four to five lines to code in any programming language. Simplicity to code is important for practitioners from other fields, since they may not be experts in programming and are looking for an algorithm that can be simply implemented and tuned to solve their domain-specific problems. Note that although PSO is also very easy to code, the performance of DE and its variants is largely better than the PSO variants over a wide variety of problems as has been indicated by studies like [21], [82], [104] and the CEC competition series [S202].
2) As indicated by the recent studies on DE [21], [82], [118] despite its simplicity, DE exhibits much better performance in comparison with several others like G3 with PCX, MA-S2, ALEP, CPSO-H, and so on of current interest on a wide variety of problems including unimodal, multimodal, separable, non-separable and so on. Although some very strong EAs like the restart CMA-ES was able to beat DE at CEC 2005 competition, on non-separable objective functions, the gross performance of DE in terms of accuracy, convergence speed, and robustness still makes it attractive for applications to various real-world optimization problems, where finding an approximate solution in reasonable amount of computational time is much weighted.
3) The number of control parameters in DE is very few ($Cr$, $F$, and $NP$ in classical DE). The effects of these parameters on the performance of the algorithm are well-studied. As will be discussed in the next section, simple adaptation rules for $F$ and $Cr$ have been devised to improve the performance of the algorithm to a large extent without imposing any serious computational burden [10], [76], [118].
4) The space complexity of DE is low as compared to some of the most competitive real parameter optimizers like CMA-ES [S232]. This feature helps in extending DE for handling large scale and expensive optimization problems. Although CMA-ES remains very competitive over problems up to 100 variables, it is difficult to extend it to higher dimensional problems due to its storage, update, and inversion operations over square matrices with size the same as the number of variables.

Perhaps these issues triggered the popularity of DE among researchers all around the globe within a short span of time as is evident from the bibliography of DE [37] from 1997 to 2002. Consequently, over the past decade research on and with DE has become huge and multifaceted. Although there exists a few significant survey papers on EAs and swarm intelligence algorithms (e.g., [S21]–[S25]), to the best of our knowledge no extensive review article capturing the entire horizon of the current DE-research has so far been published. In a recently published article [60], Neri and Tirronen reviewed a number of DE-variants for single-objective optimization problems and also made an experimental comparison of these variants on a set of numerical benchmarks. 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. This paper attempts to provide a comprehensive survey of the DE algorithm—its basic concepts, different structures, and variants for

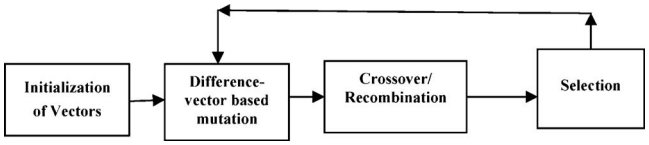



Fig. 1. Main stages of the DE algorithm.

solving constrained, multiobjective, dynamic, and large-scale optimization problems as well as applications of DE variants to practical optimization problems. The rest of this paper is arranged as follows. In Section II, the basic concepts related to classical DE are explained along with the original formulation of the algorithm in the real number space. Section III discusses the parameter adaptation and control schemes for DE. Section IV provides an overview of several prominent variants of the DE algorithm. Section V provides an extensive survey on the applications of DE to the discrete, constrained, multiobjective, and dynamic optimization problems. The theoretical analysis of DE has been reviewed in Section VI. Section VII provides an overview of the most significant engineering applications of DE. The drawbacks of DE are pointed out in Section VIII. Section IX highlights a number of future research issues related to DE. Finally, the paper is concluded in Section X.

## II. Differential Evolutions: Basic Concepts and Formulation in Continuous Real Space

Scientists and engineers from all disciplines often have to deal with the classical problem of search and optimization. Optimization means the action of finding the best-suited solution of a problem within the given constraints and flexibilities. While optimizing performance of a system, we aim at finding out such a set of values of the system parameters for which the overall performance of the system will be the best under some given conditions. Usually, the parameters governing the system performance are represented in a vector like $\vec{X} = [x_1, x_2, x_3, ..., x_D]^T$. For real parameter optimization, as the name implies, each parameter $x_i$ is a real number. To measure how far the "best" performance we have achieved, an objective function (or fitness function) is designed for the system. The task of optimization is basically a search for such the parameter vector $\vec{X}^*$, which minimizes such an objective function $f(\vec{X})(f : \Omega \subseteq \Re^D \rightarrow \Re)$, i.e., $f(\vec{X}*) < f(\vec{X})$ for all $\vec{X} \in \Omega$, where $\Omega$ is a non-empty large finite set serving as the domain of the search. For unconstrained optimization problems $\Omega = \Re^D$. Since $\max\{f(\vec{X})\} = -\min\{-f(\vec{X})\}$, the restriction to minimization is without loss of generality. In general, the optimization task is complicated by the existence of non-linear objective functions with multiple local minima. A local minimum $f_\ell = f(\vec{X}_\ell)$ may be defined as $\exists \varepsilon > 0 \, \forall \vec{X} \in \Omega : \left\| \vec{X} - \vec{X}_\ell \right\| < \varepsilon \Rightarrow f_\ell \leq f(\vec{X})$, where $\|.\|$ indicates any $p$-norm distance measure.

DE is a simple real parameter optimization algorithm. It works through a simple cycle of stages, presented in Fig. 1. We explain each stage separately in Sections II-A–II-D.

### A. Initialization of the Parameter Vectors

DE searches for a global optimum point in a $D$-dimensional real parameter space $\Re^D$. It begins with a randomly initiated population of $NP$ $D$ dimensional real-valued parameter vectors. Each vector, also known as *genome/chromosome*, forms a candidate solution to the multidimensional optimization problem. We shall denote subsequent generations in DE by $G = 0, 1..., G_{\max}$. Since the parameter vectors are likely to be changed over different generations, we may adopt the following notation for representing the $i$th vector of the population at the current generation:

$$\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, ....., x_{D,i,G}]. \tag{1}$$

For each parameter of the problem, there may be a certain range within which the value of the parameter should be restricted, often because parameters are related to physical components or measures that have natural bounds (for example if one parameter is a length or mass, it cannot be negative). The initial population (at $G = 0$) should cover this range as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum bounds: $\vec{X}_{\min} = \{x_{1,\min}, x_{2,\min}, ..., x_{D,\min}\}$ and $\vec{X}_{\max} = \{x_{1,\max}, x_{2,\max}, ..., x_{D,\max}\}$. Hence we may initialize the $j$th component of the $i$th vector as

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

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.

### B. Mutation with Difference Vectors

Biologically, "mutation" means a sudden change in the gene characteristics of a chromosome. In the context of the evolutionary computing paradigm, however, mutation is also seen as a change or perturbation with a random element. In DE-literature, a parent vector from the current generation is called *target* vector, a mutant vector obtained through the differential mutation operation is known as *donor* vector and finally an offspring formed by recombining the donor with the target vector is called *trial* vector. In one of the simplest forms of DE-mutation, to create the donor vector for each $i$th target vector from the current population, three other distinct parameter vectors, say $\vec{X}_{r_1^i}$, $\vec{X}_{r_2^i}$, and $\vec{X}_{r_3^i}$ are sampled randomly from the current population. The indices $r_1^i$, $r_2^i$, and $r_3^i$ are mutually exclusive integers randomly chosen from the range [1, $NP$], which are also different from the base vector index $i$. These indices are randomly generated once for each mutant vector. Now the difference of any two of these three vectors is scaled by a scalar number $F$ (that typically lies in the interval [0.4, 1]) and the scaled difference is added to the third one whence we obtain the donor vector $\vec{V}_{i,G}$. We can express the process as

$$\vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}). \tag{3}$$

The process is illustrated on a 2-D parameter space (showing constant cost contours of an arbitrary objective function) in Fig. 2.

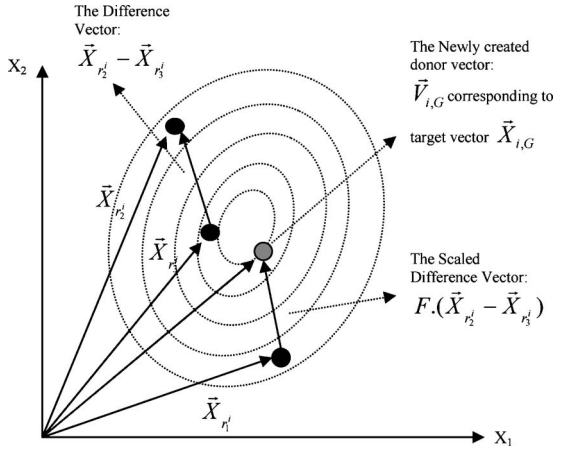



Fig. 2. Illustrating a simple DE mutation scheme in 2-D parametric space.

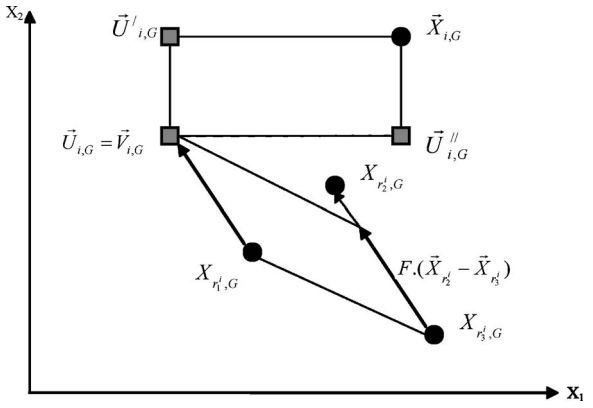



Fig. 3. Different possible trial vectors formed due to uniform/binomial crossover between the target and the mutant vectors in 2-D search space.

### C. Crossover

To enhance the potential diversity of the population, a crossover operation comes into play after generating the donor vector through mutation. The donor vector exchanges its components with the target vector $\vec{X}_{i,G}$ under this operation to form the trial vector $\vec{U}_{i,G} = [u_{1,i,G}, u_{2,i,G}, u_{3,i,G}, ..., u_{D,i,G}]$. The DE family of algorithms can use two kinds of crossover methods—*exponential* (or two-point modulo) and *binomial* (or uniform) [74]. In exponential crossover, we first choose an integer $n$ randomly among the numbers $[1, D]$. 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 choose another integer $L$ from the interval $[1, D]$. $L$ denotes the number of components the donor vector actually contributes to the target vector. After choosing $n$ and $L$ the trial vector is obtained as

$$\begin{aligned} u_{j,i,G} &= v_{j,i,G} \quad \text{for } j = \langle n \rangle_D \quad \langle n+1 \rangle_D, ..., \langle n+L-1 \rangle_D \\ x_{j,i,G} &\text{ for all other } j \in [1, D] \end{aligned} \tag{4}$$

where the angular brackets $\langle \rangle_D$ denote a modulo function with modulus $D$. The integer $L$ is drawn from $[1, D]$ according to the following pseudo-code:

$L = 0$; DO
{
$L = L + 1$;
} WHILE (($rand(0, 1) \leq Cr$) AND ($L \leq D$)).

"$Cr$" is called the *crossover rate* and appears as a control parameter of DE just like $F$. Hence in effect, probability ($L = v$) = $(Cr)v - 1$ for any positive integer $v$ lying in the interval $[1, D]$. For each donor vector, a new set of $n$ and $L$ must be chosen randomly as shown above.

On the other hand, 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 the $Cr$ value. In this case, the number of parameters inherited from the donor has a (nearly) binomial distribution. The scheme may be outlined as

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & \text{if } (rand_{i,j}[0, 1] \leq Cr \text{ or } j = j_{rand}) \\ x_{j,i,G} & \text{otherwise} \end{cases} \tag{5}$$

where, as before, $rand_{i,j}[0, 1]$ is a uniformly distributed random number, which is called anew for each $j$th component of the $i$th parameter vector. $j_{rand} \in [1, 2, ...., D]$ is a randomly chosen index, which ensures that $\vec{U}_{i,G}$ gets at least one component from $\vec{V}_{i,G}$. It is instantiated once for each vector per generation. We note that for this additional demand, $Cr$ is only approximating the true probability $p_{Cr}$ of the event that a component of the trial vector will be inherited from the donor. Also, one may observe that in a 2-D search space, three possible trial vectors may result from uniformly crossing a mutant/donor vector $\vec{V}_{i,G}$ with the target vector $\vec{X}_{i,G}$. These trial vectors are as follows.

1) $\vec{U}_{i,G} = \vec{V}_{i,G}$ such that both the components of $\vec{U}_{i,G}$ are inherited from $\vec{V}_{i,G}$.
2) $\vec{U}^{I}_{i,G}$, in which the first component ($j = 1$) comes from $\vec{V}_{i,G}$ and the second one ($j = 2$) from $\vec{X}_{i,G}$.
3) $\vec{U}^{II}_{i,G}$, in which the first component ($j = 1$) comes from $\vec{X}_{i,G}$ and the second one ($j = 2$) from $\vec{V}_{i,G}$.

The possible trial vectors due to uniform crossover are illustrated in Fig. 3.

### D. Selection

To keep the population size constant over subsequent generations, the next step of the algorithm calls for *selection* to determine whether the target or the trial vector survives to the next generation, i.e., at $G = G + 1$. The selection operation is described as

$$\begin{aligned} \vec{X}_{i,G+1} &= \vec{U}_{i,G} \quad \text{if } f(\vec{U}_{i,G}) \leq f(\vec{X}_{i,G}) \\ &= \vec{X}_{i,G} \quad \text{if } f(\vec{U}_{i,G}) > f(\vec{X}_{i,G}) \end{aligned} \tag{6}$$

where $f(\vec{X})$ 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 generation; otherwise the target is retained in the population. Hence, the population either gets better (with respect to the minimization of the objective function) or remains the same in fitness status, but never deteriorates. Note that in (6) the target vector is replaced by the trial vector even if both yields the same value of the objective function—a feature that enables DE-vectors to move over

**Algorithm 1** Pseudo-code for the DE Algorithm with Binomial Crossover

**Step 1:** Read values of the control parameters of DE: scale factor $F$, crossover rate $Cr$, and the population size $NP$ from user.

**Step 2:** Set the generation number $G = 0$ and randomly initialize a population of $NP$ individuals $P_G = \{\vec{X}_{1,G}, ....., \vec{X}_{NP,G}\}$ with $\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, ....., x_{D,i,G}]$ and each individual uniformly distributed in the range $[\vec{X}_{\min}, \vec{X}_{\max}]$, where $\vec{X}_{\min} = \{x_{1,\min}, x_{2,\min}, ..., x_{D,\min}\}$ and $\vec{X}_{\max} = \{x_{1,\max}, x_{2,\max}, ..., x_{D,\max}\}$ with $i = [1, 2, ...., NP]$.

**Step 3**. WHILE the stopping criterion is not satisfied DO

FOR $i = 1$ to $NP$ //do for each individual sequentially

**Step 2.1** ***Mutation Step***

Generate a donor vector $\vec{V}_{i,G} = \{v_{1,i,G}, ....., \}$ $\{v_{D,i,G}\}$ corresponding to the $i$th target vector $\vec{X}_{i,G}$ via the differential mutation scheme of DE as:

$$\vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}).$$

**Step 2.2** ***Crossover Step***

Generate a trial vector $\vec{U}_{i,G} = \{u_{1,i,G}, ....., u_{D,i,G}\}$ for the $i$th target vector $\vec{X}_{i,G}$ through binomial crossover in the following way:

$u_{j,i,G} = v_{j,i,G}$, if $(rand_{i,j}[0, 1] \leq Cr$ or $j = j_{rand})$
$x_{j,i,G}$, otherwise,

**Step 2.3** ***Selection Step***

Evaluate the trial vector $\vec{U}_{i,G}$

IF $f(\vec{U}_{i,G}) \leq f(\vec{X}_{i,G})$, THEN $\vec{X}_{i,G+1} = \vec{U}_{i,G}$

ELSE $\vec{X}_{i,G+1} = \vec{X}_{i,G}$.

END IF

END FOR

**Step 2.4** Increase the Generation Count $G = G + 1$

END WHILE

flat fitness landscapes with generations. Note that throughout this paper, we shall use the terms *objective function value* and *fitness* interchangeably. But, always for minimization problems, a lower objective function value will correspond to higher fitness.

### *E. Summary of DE Iteration*

An iteration of the classical DE algorithm consists of the four basic steps—initialization of a population of search variable vectors, mutation, crossover or recombination, and finally selection. After having illustrated these stages, we now formally present the whole of the algorithm in a pseudo-code below.

The terminating condition can be defined in a few ways like: 1) by a fixed number of iterations $G_{max}$, with a suitably large value of $G_{max}$ depending upon the complexity of the objective function; 2) when best fitness of the population does not change appreciably over successive iterations; and alternatively 3) attaining a pre-specified objective function value.

### *F. DE Family of Storn and Price*

Actually it is the process of mutation that demarcates one DE scheme from another. In the previous section, we have illustrated the basic steps of a simple DE. The mutation scheme in (3) uses a randomly selected vector $\vec{X}_{r1}$ and only one weighted difference vector $F \cdot (\vec{X}_{r2} - \vec{X}_{r3})$ to perturb it. Hence, in literature, the particular mutation scheme given by (3) is referred to as DE/rand/1. When used in conjunction with binomial crossover, the procedure is called DE/rand/1/bin. We can now have an idea of how the different DE schemes are named. The general convention used above is DE/x/y/z, where DE stands for "differential evolution," x represents a string denoting the base vector to be perturbed, y is the number of difference vectors considered for perturbation of x, and z stands for the type of crossover being used (exp: exponential; bin: binomial). The other four different mutation schemes, suggested by Storn and Price [74], [75] are summarized as

$$\text{"}DE/best/1\text{:"}\ \vec{V}_{i,G} = \vec{X}_{best,G} + F \cdot (\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}) \quad (7)$$

$$\text{"}DE/target - to - best/1\text{:"}\ \vec{V}_{i,G} = \vec{X}_{i,G} + F \cdot (\vec{X}_{best,G} - \vec{X}_{i,G}) + F \cdot (\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}) \quad (8)$$

$$\text{"}DE/best/2\text{:"}\ \vec{V}_{i,G} = \vec{X}_{best,G} + F \cdot (\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}) + F \cdot (\vec{X}_{r_3^i,G} - \vec{X}_{r_4^i,G}) \quad (9)$$

$$\text{"}DE/rand/2\text{:"}\ \vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}) + F \cdot (\vec{X}_{r_4^i,G} - \vec{X}_{r_5^i,G}). \quad (10)$$

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 once for each donor vector. The scaling factor $F$ is a positive control parameter for scaling the difference vectors. $\vec{X}_{best,G}$ is the best individual vector with the best fitness (i.e., lowest objective function value for a minimization problem) in the population at generation $G$. Note that some of the strategies for creating the donor vector may be mutated recombinants, for example, (8) listed above basically mutates a two-vector recombinant $\vec{X}_{i,G} + F \cdot (\vec{X}_{best,G} - \vec{X}_{i,G})$.

Storn and Price [74], [92] suggested a total of ten different working strategies for DE and some guidelines in applying these strategies to any given problem. These strategies were derived from the five different DE mutation schemes outlined above. Each mutation strategy was combined with either the "exponential" type crossover or the "binomial" type crossover. This yielded a total of $5 \times 2 = 10$ DE strategies. In fact many other linear vector combinations can be used for mutation. In general, no single mutation method [among those described in (3), (7)–(10)] has turned out to be best for all problems.

Nevertheless the various mutation schemes need further investigation to determine under which circumstances they perform well and on what kind of problems they yield poor results. Some initial work in this direction was undertaken by Mezura-Montes *et al.*, who empirically compared eight different DE-schemes over a test-suite of 13 benchmark problems in [56]. The authors took into account an interesting mutation scheme known as DE/rand/2/dir [26] that incorporates the objective function information to guide the direction of the donor vectors in the following way:

$$\vec{V}_{i,G} = \vec{X}_{r_1,G} + \frac{F}{2} \cdot (\vec{X}_{r_1,G} - \vec{X}_{r_2,G} - \vec{X}_{r_3,G}) \qquad (11)$$

where $\vec{X}_{r_1,G}$, $\vec{X}_{r_2,G}$, and $\vec{X}_{r_3,G}$ are distinct population members such that $f(\vec{X}_{r_1,G}) \leq \{f(\vec{X}_{r_2,G}), f(\vec{X}_{r_3,G})\}$. The experiments performed by Mezura-Montes *et al.* indicate that DE/best/1/bin (using always the best solution to find search directions and also binomial crossover) remained the most competitive scheme, regardless the characteristics of the problem to be solved, based on final accuracy and robustness of results. The authors in [56] also mention that over unimodal and separable functions, DE/rand/2/dir achieved considerably good results. For unimodal and non-separable functions, DE/best/1/bin consistently yielded best performance. This variant was also successful in optimizing the multimodal and separable benchmarks. DE/rand/1/bin and DE/rand/2/dir provided performances of similar quality on this class of functions. However, on multimodal and non-separable functions DE/rand/2/dir remained most competitive and slightly faster to converge to the global optimum.

### *G. DE and the Contemporary EAs: Conceptual Similarities and Differences*

In this section, we briefly discuss how DE relates to and differs from the contemporary EAs for real parameter optimization. Following the convention of ES, we shall use $\mu$ to indicate the number of parent vectors and $\lambda$ ($\geq \mu$) to denote the size of the child population.

1) *Mutation:* In the context of GAs and EAs, mutation is treated as a random change of some parameter. Real valued EAs typically simulate the effects of mutation with additive increments that are randomly generated by a predefined and fixed probability density function (PDF). DE differs markedly from algorithms like ES and EP in consideration of the fact that it mutates the base vectors (secondary parents) with scaled population-derived difference vectors. As generations pass, these differences tend to adapt to the natural scaling of the problem. For example, if the population becomes compact in one variable but remains widely dispersed in another, the difference vectors sampled from it will be small in the former variable, yet large in the latter. This automatic adaptation significantly improves the convergence of the algorithm. In other words, ES and EP require the specification or adaptation of absolute step size for each variable over generations while DE requires only the specification of a single relative scale factor $F$ for all variables.

Although the difference vector based mutation is believed to be one of the main strength of DE [73], [92], the idea of using difference of population members in recombination of EAs is not completely new. Eshelman and Schaffer [S187] came up with an idea of a difference-based recombination operator [called blend crossover operator (BLX)] for real coded GAs, long back in 1992. Voigt *et al.* [S188] used a *selection differential* defined as the difference between the mean fitness of the selected parents and the mean fitness of the population to derive a design criteria for recombination operators and used it with the fuzzy recombination (FR). In 1995, Deb and Agrawal [S189] proposed a simulated binary crossover (SBX) that works with two parent solutions and creates two offspring solutions to simulate the working principle of the single-point crossover operator on binary strings. In SBX, the probability distribution used to create offspring depends on a spread factor that is defined as ratio of the absolute difference in children values to that of the parent values. Both BLX and SBX were analyzed in detail by Deb and Beyer in [8] and [24]. In Kita *et al.*'s uniform normal distribution crossover (UNDX) [S190] and simplex crossover (SPX) [S191], operators generate $\mu - 1$ direction vectors for $\mu - 1$ randomly chosen parents by taking the difference of each parent vector from their mean vector. Using the direction vectors, in UNDX the probability of creating the offspring away from the mean vector is reduced and a maximum probability is assigned at the mean vector. SPX assigns a uniform probability for creating any offspring within a restricted region (called the simplex). In Deb *et al.*'s parent centric crossover (PCX) operator [S192] for each offspring one parent is chosen randomly and a difference vector is calculated between the parent and the mean of the chosen $\mu$ parents. However, the use of scaled difference of any two distinct population members to perturb a third one, as done in DE, finds closest resemblance with the reflection operation of Nelder-Mead polyhedron search [S19] and Price's CRS algorithm [S20]. Although due to space limitations, it is not possible to discuss these two algorithms in sufficient details, for interested readers we would like to point out that unlike DE, the Nelder-Mead algorithm restricts the number of sample vectors (from which the difference vector is to be generated) to $D + 1$ ($D$ corresponding to the dimensionality of the search space). This limitation becomes a drawback for complicated objective functions that require many more points to form a clear model of the surface topography. Also both Nelder-Mead's and CRS's reflection operations with difference vectors are a form of arithmetic recombination, while DE's difference vector based perturbation schemes more closely resemble a mutation operation [74, p. 29]. One of the most fundamental aspects of DE-type mutation is the fact that vector perturbations are generated from the $NP \cdot (NP - 1)$ nonzero difference vectors of the population rather than employing a predetermined PDF. This leads to one of the main assets of DE: *contour matching*, a term coined and explained by Price *et al.* in [74]. Contour matching refers to the phenomena of adaptation of the vector population such that promising regions of the fitness landscape are investigated automatically once they are detected. One of the biggest advantages that the difference vectors afford is that both a mutation step's size and its orientation are automatically adapted to the objective function landscape. Price *et al.* claim that contour matching

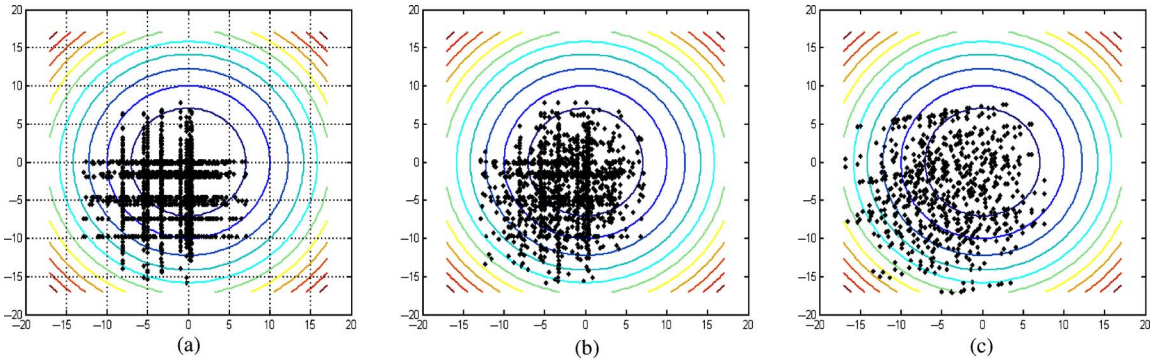



Fig. 4. Empirical distributions of candidate trial vectors for three different $Cr$ values. (a) $Cr = 0$. (b) $Cr = 0.5$. (c) $Cr = 1.0$.

also induces an important ingredient besides selection is the promotion of *basin-to-basin* transfer, where search points may move from one basin of attraction, i.e., a local minimum, to another one.

In PSO also the stochastic attraction toward the personal best and neighborhood best positions are modeled by scaled difference vectors. The velocity update formula of PSO has similarities with the DE/target-to-best/1 scheme [see (8)] that generates a mutated recombinant.

2) *Crossover:* Both DE and ES employ crossover to create a single trial vector, while most GAs recombine two vectors to produce two trial vectors often by one-point crossover. Note that EP depends only on mutation to generate offspring and does not have any crossover operator associated with. One of the popular crossover techniques for real coded GAs is the $n$-point crossover where the offspring vector is randomly partitioned into $(n + 1)$ blocks such that parameters in adjacent partitions are inherited from different parent vectors. Studies of $n$-point crossover [S193] indicate that an even number of crossover points reduces the representational bias (dependence of ordering of parameters within a vector) at the cost of increasing the disruption of parameters that are closely grouped. As analyzed by Price *et al.* [92, p. 93] DE's exponential crossover employs both one and two point crossover with an objective of reducing their individual biases. The representational bias inherent in $n$-point crossover can be eliminated if donors are determined by $D$ independent random trials. This procedure is known as uniform crossover in EA literature [S3] and this is exactly what DE employs as discrete recombination or binomial crossover [see (5)] most often.

3) *Selection:* Selection can be applied to an evolutionary process in primarily two different stages—first stage being *parent selection* to decide which vectors from the current population will undergo recombination while the second is *survivor selection* to choose which vectors from the parent and offspring populations will survive to the next generation. Unlike GAs that select parents based on their fitness, both ES and DE gives all the individuals equal chance for being selected as parents. In ES, each individual has the same chance to be selected for mutation (and recombination). In DE also the base vectors are randomly picked up without any regard for their fitness values. When only the offspring vectors are allowed to advance (as done in some simple GAs [S3]) there is no guarantee that the best-so-far solution will not be lost. Retaining the best-so-far solution is called *elitism* and it plays an important role in bringing the convergence of the algorithm to the global optimum [S193] in long time limits. For this reason and because of the speed improvement it offers, most EAs including DE, EP, and some versions of ES take into account the current population while determining the membership of the next generation.

The $(\mu, \lambda)$ ES selects best $\mu$ children to become parents in next generation. Alternatively, the $(\mu + \lambda)$ ES populates the next generation with best $\mu$ vectors from the combined parent and child populations. The survivor selection scheme of DE is closer in spirit to the elitist $(\mu + \lambda)$ ES, however, instead of ranking the combined population, the former employs a one-to-one competition where each parent vector competes once only against its own offspring. Evidently, unlike the tournament selection in EP, DE's one-to-one selection holds only $NP$ knock-out competitions between a parent and its offspring generated through mutation and recombination. Comparing each trial vector (offspring) to the best performing vectors at the same index ensures that DE retains the very best-so-far solution at each index. Parent-offspring competition has a superior ability to maintain population diversity when compared with ranking or tournament selection where elites and their offspring may dominate the population rapidly.

## III. Control Parameters of the Differential Evolution

There are three main control parameters of the DE algorithm: the mutation scale factor $F$, the crossover constant $Cr$, and the population size $NP$. In this section, we focus on the effect of each of these parameters on the performance of DE as well as the state-of-the-art methods for tuning these parameters. A good volume of research work has been undertaken so far to improve the ultimate performance of DE by tuning its control parameters. Storn and Price in [88] have indicated that a reasonable value for $NP$ could be chosen between 5-D and 10-D (D being the dimensionality of the problem), and a good initial choice of $F$ was 0.5. The effective range of $F$ is usually between 0.4 and 1.

The parameter $Cr$ controls how many parameters in expectation are changed in a population member. For low value

of $Cr$, a small number of parameters are changed in each generation and the stepwise movement tends to be orthogonal to the current coordinate axes. On the other hand, high values of $Cr$ (near 1) cause most of the directions of the mutant vector to be inherited prohibiting the generation of axis orthogonal steps. This effect has been illustrated in Fig. 4 by showing for three values of $Cr$, an empirical distribution of candidate trial vectors obtained by running DE on a single starting population of ten vectors for 200 generations with selection disabled.

It is interesting to note at this point that for algorithms like classic DE (DE/rand/1/bin) performance is rotationally invariant only when $Cr = 1$. At that setting, crossover is a vector-level operation that makes the trial vector a pure mutant, i.e., $\vec{U}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G})$. The location (with respect to the function's topography) of mutant trial vectors will not change under coordinate rotation as long as $Cr = 1$ and $F$ is a constant, or sampled from a distribution no more than once per trial vector. A low $Cr$ value (e.g., 0 or 0.1) results in a search that changes each direction (or a small subset of directions) separately. This is an effective strategy for functions that are separable or decomposable [i.e., $f(\vec{X}) = \sum_{i=1}^{D} f_i(x_i)$]

Gamperle *et al.* [S26] evaluated different parameter settings for DE on the Sphere, Rosenbrock's, and Rastrigin's functions. Their experimental results revealed that the global optimum searching capability and the convergence speed are very sensitive to the choice of control parameters $NP$, $F$, and $Cr$. Furthermore, a plausible choice of the population size $NP$ is between 3-D and 8-D, the scaling factor $F = 0.6$, and the crossover rate $Cr$ is between [0.3, 0.9]. Recently, the authors in [85] state that typically $0.4 < F < 0.95$ with $F = 0.9$ can serve as a good first choice. They also opine that $Cr$ should lie in (0, 0.2) when the function is separable, while in (0.9, 1) when the function's parameters are dependent.

As can be perceived from the literature, several claims and counter-claims were reported concerning the rules for choosing the control parameters and these can potentially confuse engineers, who may try to solve practical problems with DE. Further, most of these claims lack sufficient experimental justifications. Some objective functions are very sensitive to the proper choice of the parameter settings in DE [S27]. Therefore, researchers naturally started to consider some techniques such as self-adaptation to automatically find an optimal set of control parameters for DE [S28], [3], [10], [48], [76], [84]. Usually self-adaptation is applied to tune the control parameters $F$ and $Cr$. Liu and Lampinen [48] introduced a Fuzzy adaptive differential evolution using fuzzy logic controllers whose inputs incorporate the relative function values and individuals of successive generations to adapt the parameters for the mutation and crossover operation. In this context, Qin *et al.* [76] came up with a SaDE algorithm, in which both the trial vector generation strategies and their associated control parameters $F$ and $Cr$ are gradually self-adapted by learning from their previous experiences of generating promising solutions. The parameter $F$, in SaDE, is approximated by a normal distribution with mean value 0.5 and standard deviation 0.3, denoted by $N$ (0.5, 0.3). A set of $F$ values are randomly sampled from such normal distribution and applied to each target vector in the current population. This way, SaDE attempts to maintain both exploitation (with small $F$ values) and exploration (with large $F$ values) power throughout the entire evolution process. SaDE gradually adjusts the range of $Cr$ values for a given problem according to previous $Cr$ values that have generated trial vectors successfully entering the next generation. Specifically, it is assumed that $Cr$ obeys a normal distribution with mean value $Cr_m$ and standard deviation $Std = 0.1$, denoted by $N(Cr_m, Std)$, where $Cr_m$ is initialized as 0.5. The $Std$ should be set as a small value to guarantee that most $Cr$ values generated by $N(Cr_m, Std)$ are between [0, 1], even when $Cr_m$ is near 0 or 1. Hence, the value of $Std$ is set as 0.1. Note that the self-adaptive schemes like SaDE often themselves have parameters to be adjusted like the standard deviation in normal distribution. However, self-adaptive DE performs better than the standard DE because sensitive parameters in DE are replaced by less sensitive parameters in self-adaptive DE.

In [3], a fitness-based adaptation has been proposed for $F$. A system with two evolving populations has been implemented. The crossover rate $Cr$ has been fixed to 0.5 after an empirical study. Unlike $Cr$, the value of $F$ is adaptively updated at each generation by means of the following scheme:

$$F = \begin{cases} \max\left\{l_{\min}, 1 - \left|\frac{f_{\max}}{f_{\min}}\right|\right\} & \text{if } \left|\frac{f_{\max}}{f_{\min}}\right| < 1 \\ \max\left\{l_{\min}, 1 - \left|\frac{f_{\min}}{f_{\max}}\right|\right\} & \text{otherwise} \end{cases} \tag{12}$$

where $l_{\min} = 0.4$ is the lower bound of $f$, $f_{\min}$ and $f_{\max}$ are the minimum and maximum objective function values over the individuals of the populations, obtained in a generation.

Recently, Brest *et al.* [10] proposed a self-adaptation scheme for the DE control parameters. They encoded control parameters $F$ and $Cr$ into the individual and adjusted them by introducing two new parameters $\tau_1$ and $\tau_2$. In their algorithm (called "jDE"), a set of $F$ and $Cr$ values was assigned to each individual in the population, augmenting the dimensions of each vector. The better values of these encoded control parameters lead to better individuals that in turn, are more likely to survive and produce offspring and, thus, propagate these better parameter values. The new control parameters for the next generation are computed as follows:

$$F_{i,G+1} = \begin{cases} F_l + rand_1 * F_u & \text{with probability } \tau_1 \\ F_{i,G} & \text{else} \end{cases} \tag{13a}$$

$$\text{and} \quad Cr_{i,G+1} = \begin{cases} rand_3 & \text{with probability } \tau_2 \\ Cr_{i,G} & \text{else} \end{cases} \tag{13b}$$

where $F_l$ and $F_u$ are the lower and upper limits of $F$ and both lie in [0, 1]. In [10] and [S231], Brest *et al.* used $\tau_1 = \tau_2 = 0.1$. As $F_l = 0.1$ and $F_u = 0.9$, the new $F$ takes a value from [0.1, 0.9] while the new $Cr$ takes a value from [0, 1]. As $F_{i,G+1}$ and $CR_{i,G+1}$ values are obtained before the mutation is performed, they influence the mutation, crossover, and selection operations for the new vector $\vec{X}_{i,G+1}$.

Zaharie [S29] proposed a parameter adaptation strategy for DE (ADE) based on the idea of controlling the population diversity, and implemented a multipopulation approach. Following the same line of thinking, Zaharie and Petcu [S30]

designed an adaptive Pareto DE algorithm for multiobjective optimization and also analyzed its parallel implementation. Abbass [1] self-adapted the crossover rate $Cr$ for multiobjective optimization problems, by encoding the value of $Cr$ into each individual, simultaneously evolved with other search variables. The scaling factor $F$ was generated for each variable from a Gaussian distribution $N(0, 1)$. The upper limit of the scale factor $F$ is empirically taken as 1. Although it does not necessarily mean that a solution is not possible with $F > 1$, however, until date, no benchmark function that was successfully optimized with DE required $F > 1$. Zaharie [112] derived a lower limit of $F$ and the study [112] revealed that if $F$ is sufficiently small, the population can converge even in the absence of selection pressure. With a few simplifying assumptions, Zaharie proved the following relation between the variance of the original population $P_{x,t}$ at time step $G$ and the variance of the trial population $P_{u,t}$

$$E(Var(P_{u,t})) = \left(2 \cdot F^2 \cdot p_{Cr} - \frac{2 \cdot p_{Cr}}{NP} + \frac{p_{Cr}^2}{NP} + 1\right) \cdot Var(P_{x,t}) \tag{14}$$

where $p_{Cr}$ is the probability of crossover [Zaharie neglected the $j_{rand}$ part in (5) and then $Cr$ became the absolute probability that a component of the target vector is exchanged with that of the donor vector; Zaharie used the notation $p_{Cr}$ to denote this probability instead of $Cr$]. Consequently, the DE control parameter combinations that satisfy the equation

$$2.F^2 - \frac{2}{NP} + \frac{p_{Cr}}{NP} = 0 \tag{15}$$

may be considered as critical since they result in a population whose variance remains constant except for random fluctuations. Thus, when the selection step is absent, according to (13), $F$ will display a critical value $F_{crit}$ such that the population variance decreases when $F < F_{crit}$ and increases if $F > F_{crit}$. Solving (13), we have

$$F_{crit} = \sqrt{\frac{\left(1 - p_{Cr}/2\right)}{NP}}. \tag{16}$$

Zaharie experimentally confirmed that $F_{crit}$ establishes a lower limit on the value of $F$ in the sense that smaller values will induce convergence even on a flat objective function landscape (when all trial vectors are accepted, i.e., selection pressure is absent). Omran *et al*. [65] introduced a self-adaptive scaling factor parameter $F$. They generated the value of $Cr$ for each individual from a normal distribution $N(0.5, 0.15)$. This approach (called "SDE") was tested on four benchmark functions and performed better than other versions of DE. Besides adapting the control parameters $F$ or $Cr$, some researcher also adapted the population size. Teo [102] proposed DE with self-adaptive population size $NP$ (abbreviated as DESAP), based on self-adaptive Pareto DE proposed by Abbas [1].

Mallipeddi and Suganthan [50] empirically investigated the effect of population size on the quality of solutions and the computational effort required by DE with a set of five problems chosen from the test-suite of CEC 2005 Special Session on Real-Parameter Optimization [95]. In [11], the authors presented a method for gradually reducing population size of DE. The method improves the efficiency and robustness of the algorithm and can be applied to any variant of a DE algorithm. In [51], Mallipeddi and Suganthan proposed a DE algorithm with an ensemble of parallel populations, where the number of function evaluations (FEs) allocated to each population is self-adapted by learning from their previous experiences in generating superior solutions. Consequently, a more suitable population size along with its parameter settings can be determined adaptively to match different search/evolution phases.

Apart from self-adaptation, frequently $F$ has been made to vary randomly for improving the performances of DE. Price *et al*. [74] defined two new terms: *jitter* and *dither* in context to the randomization of $F$. The practice of generating a new value of $F$ for every *parameter* is called *jitter* and it is signified by subscripting $F$ with the parameter index, $j$. Alternatively, choosing $F$ anew for each *vector*, or *dithering*, is indicated by subscripting $F$ with the population's running index, $i$. Dithering scales the length of vector differentials because the same factor, $F_i$, is applied to all components of a difference vector. Das *et al*. used dither in [17] where $F$ was made to vary randomly between 0.5 and 1 for each vector. In the same paper, they also suggested decreasing $F$ linearly from 1.0 to 0.5 in their second scheme (called DETVSF: DE with time varying scale factor). This encourages the individuals to sample diverse zones of the search space during the early stages of the search (promoting exploration). During the later stages a decaying scale factor helps to adjust the movements of trial solutions finely so that they can explore the interior of a relatively small space in which the suspected global optimum lies (thus promoting exploitation). Recently in works like [S31] and [S32], chaotic sequences are combined with DE in order to enhance its population diversity and thus to avoid the state of stagnation, when a standard DE may occasionally stop proceeding toward the global optimum virtually without any obvious reasons [41]. That means although the population has not converged to a local optimum or any other point, the population is still remaining diverse, and occasionally, even new individuals may enter the population, but the algorithm does not progress by finding any better solutions. Chaos theory [S33] deals with the qualitative study of unstable aperiodic behavior in deterministic nonlinear dynamical systems. In chaotic DE the scale factor $F$ is varied over generations by using the logistic map iterator, which is one of the simplest dynamic systems evidencing chaotic behavior, in the following way:

$$F_G = \mu \cdot F_{G-1} \cdot [1 - F_{G-1}]. \tag{17}$$

## IV. Important Variants of DE for Continuous Single-Objective Optimization

Since its advent in 1995, DE has been attracting the attention of the researchers from diverse domains of knowledge, all over the world. This has resulted in a wealth of variants of the basic DE algorithm. Some of these variants are devised to tackle specific applications while others are generalized for numerical optimization. In this section, we shall undertake

an in-depth discussion of the most prominent DE-variants that were developed over the past decade and appeared to be competitive against the existing best-known real parameter optimizers.

### A. Differential Evolution Using Trigonometric Mutation

Fan and Lampinen [25] proposed a trigonometric mutation operator for DE to speed up its performance. To implement the scheme, for each target vector, three distinct vectors are randomly selected from the DE population. Suppose for the $i$th target vector $\vec{X}_{i,G}$, the selected population members are $\vec{X}_{r_1,G}$, $\vec{X}_{r_2,G}$, and $\vec{X}_{r_3,G}$. The indices $r_1$, $r_2$, and $r_3$ are mutually exclusive integers randomly chosen from the range [1, $NP$], which are also different from the index $i$. Now three weighting coefficients are formed according to the following equations:

$$p' = \left|f(\vec{X}_{r_1})\right| + \left|f(\vec{X}_{r_2})\right| + \left|f(\vec{X}_{r_3})\right| \tag{18a}$$

$$p_1 = \left|f(\vec{X}_{r_1})\right| / p' \tag{18b}$$

$$p_2 = \left|f(\vec{X}_{r_2})\right| / p' \tag{18c}$$

$$p_3 = \left|f(\vec{X}_{r_3})\right| / p' \tag{18d}$$

where $f()$ is the function to be minimized. Let $\Gamma$ be the trigonometric mutation rate in the interval (0, 1). Then the trigonometric mutation scheme may now be expressed as

$$\vec{V}_{i,G+1} = (\vec{X}_{r1} + \vec{X}_{r2} + \vec{X}_{r3})/3 + (p_2 - p_1).(\vec{X}_{r1} - \vec{X}_{r2}) + (p_3 - p_2) \cdot (\vec{X}_{r2} - \vec{X}_{r3}) + (p_1 - p_3) \cdot (\vec{X}_{r3} - \vec{X}_{r1}) \text{ if } rand[0, 1] \leq \Gamma$$

$$\vec{V}_{i,G+1} = \vec{X}_{r1} + F \cdot (\vec{X}_{r2} - \vec{X}_{r3}) \qquad \text{else.} \tag{19}$$

Thus, the scheme proposed by Fan *et al*. used trigonometric mutation with a probability of $\Gamma$ and the mutation scheme of DE/rand/1 with a probability of $(1 - \Gamma)$.

### B. Differential Evolution Using Arithmetic Recombination

The binomial crossover scheme, usually employed in most of the DE variants, creates new combinations of parameters; it leaves the parameter values themselves unchanged. Binomial crossover is in spirit same as the discrete recombination used in conjunction with many EAs. However, in *continuous* or *arithmetic* recombination, the individual components of the trial vector are expressed as a linear combination of the components from mutant/donor vector and the target vector. The common form of the arithmetic recombination between two vectors $\vec{X}_{r_1,G}$ and $\vec{X}_{r_2,G}$ adopted by most of the EAs [S3] may be put as

$$\vec{W}_{i,G} = \overrightarrow{X}_{r_1,G} + k_i \cdot (\vec{X}_{r_1,G} - \vec{X}_{r_2,G}). \tag{20}$$

The *coefficient of combination* $k_i$ can either be a constant or a random variable. Generally speaking, if this coefficient is sampled anew for each vector then the resulting process is known as line recombination. However, if the combination coefficient is elected randomly anew for each component of the vectors to be crossed, then the process is known as intermediate recombination and may be formalized for the $j$th component of the recombinants as

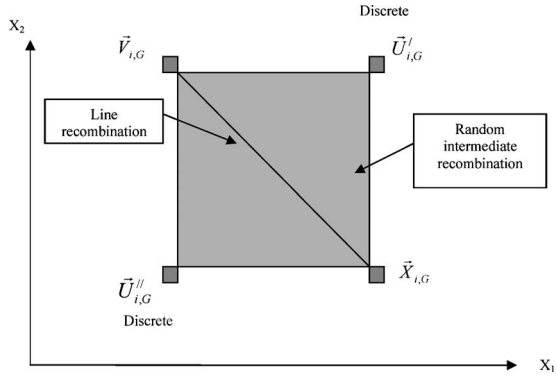



Fig. 5. Domains of the different recombinant vectors generated using discrete, line and random intermediate recombination.

$$w_{i,j,G} = x_{r_1,j,G} + k_j \cdot (x_{r_1,j,G} - x_{r_2,j,G}). \tag{21}$$

Fig. 5 schematically shows the regions searched by discrete, line and arithmetic recombination between donor vector $\vec{V}_{i,G}$ and the target vector $\vec{X}_{i,G}$ when the coefficient of combination is a uniformly distributed random number between 0 and 1. The two recombinant vectors occupy the opposite corners of a hypercube whose remaining corners are the trial vectors $\vec{U}'_{i,G}$ and $\vec{U}''_{i,G}$ created by discrete recombination. Line recombination, as its name suggests, searches along the axis connecting the recombinant vectors, while the intermediate recombination explores the entire $D$-dimensional volume contained within the hypercube. As can be perceived from Fig. 5, both the discrete as well as the intermediate recombination are not rotationally invariant processes. If the coordinate system rotates through an angle, the corners of the hypercube are relocated, which in turn redefines the area searched by the intermediate recombination. On the other hand, the line recombination is rotationally invariant.

To make the recombination process of DE rotationally invariant, Price proposed a new trial vector generation strategy "DE/current-to-rand/1" [75], which replaces the binomial crossover operator with the rotationally invariant arithmetic line recombination operator to generate the trial vector $\vec{U}_{i,G}$ by linearly combining the target vector $\vec{X}_{i,G}$ and the corresponding donor vector $\vec{V}_{i,G}$ as follows:

$$\vec{U}_{i,G} = \vec{X}_{i,G} + k_i \cdot (\vec{V}_{i,G} - \vec{X}_{i,G}). \tag{22}$$

Now incorporating (3) in (22) we have

$$\vec{U}_{i,G} = \vec{X}_{i,G} + k_i \cdot (\vec{X}_{r_1,G} + F \cdot (\vec{X}_{r_2,G} - \vec{X}_{r_3,G}) - \vec{X}_{i,G}) \tag{23}$$

which further simplifies to

$$\vec{U}_{i,G} = \vec{X}_{i,G} + k_i \cdot (\vec{X}_{r_1,G} - \vec{X}_{i,G}) + F' \cdot (\vec{X}_{r_2,G} - \vec{X}_{r_3,G}) \tag{24}$$

where $k_i$ is the combination coefficient, which has been experimentally shown [74], [75] to be effective when it is chosen with a uniform random distribution from [0, 1] and $F' = k_i \cdot F$ is a new constant parameter.

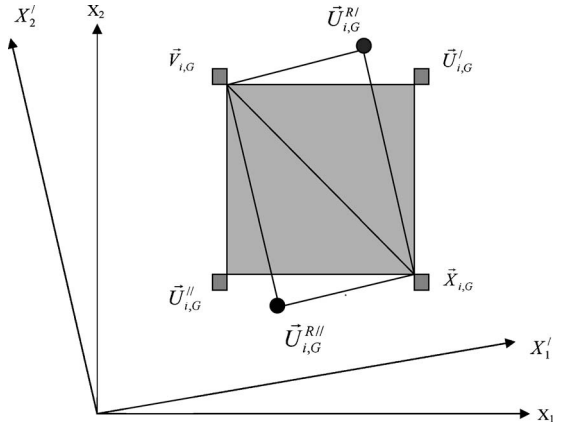



Fig. 6. Change of the trial vectors generated through the discrete and random intermediate recombination due to rotation of the coordinate system. $\vec{U}_{i,G}^{R/}$ and $\vec{U}_{i,G}^{R//}$ indicate the new trial vectors due to discrete recombination in rotated coordinate system.

### C. DE/rand/1/Either-Or Algorithm

Price *et al.* [74, p. 118] proposed the state-of-the-art DE/rand/1/either-or algorithm, where the trial vectors that are pure mutants occur with a probability $p_F$ and those that are pure recombinants occur with a probability $1 - p_F$. This variant is shown to yield competitive results against classical DE-variants rand/1/bin and target-to-best/1/bin in a recent comparative study [21]. The scheme for trial vector generation may be outlined as

$$\vec{U}_{i,G} = \begin{cases} \vec{X}_{r_1,G} + F.(\vec{X}_{r_2,G} - \vec{X}_{r_3,G}) & \text{if } rand_i(0,1) < p_F \\ = \vec{X}_{r_0,G} + k.(\vec{X}_{r_1} + \vec{X}_{r_2} - 2.\vec{X}_{r_0}) & \text{otherwise} \end{cases}. \quad (25)$$

Price *et al.* recommended $k = 0.5 \cdot (F + 1)$ as a good choice for the parameter $k$ for a given $F$. The DE/rand/1/either-or algorithm provides a simple way to implement the dual axis search in the $k$-$F$ plane ($k$ indicating the combination coefficient of the arithmetic crossover and $F$ being the scale factor). The scheme provides efficient solutions for functions that are best minimized by either mutation-only ($p_F = 1$) or recombination only ($p_F = 0$), as well as generic functions that can be solved by randomly interleaving both operations ($0 < p_F < 1$). Note that $p_F$ is a parameter of the algorithm and it determines the relative importance of the mutation and arithmetic recombination schemes. Price *et al.* recommend a value 0.4 for it. It is interesting to investigate whether it is possible to self-adapt $p_F$ so that the algorithm may be able to decide the optimal value of this parameter capturing some special properties of the objective function under test.

### D. Opposition-Based Differential Evolution

The concept of opposition-based learning was introduced by Tizhoosh [S34] and its applications were introduced in [S34]–[S36]. Rahnamayan *et al.* [82] have recently proposed an ODE for faster global search and optimization. The algorithm also finds important applications to the noisy optimization problems [S37]. The conventional DE was enhanced by utilizing *opposition number* based optimization concept in three levels, namely, population initialization, generation jumping, and local improvement of the population's best member. In the absence of *a priori* information about the actual optima, an EA usually starts with *random guesses*. We can improve our chance of starting with a better solution by simultaneously checking fitness of *the opposite solution*. By doing this, the fitter one (guess or opposite guess) can be chosen as an initial solution. As explained in [S34], according to probability theory, 50% of the time a guess may have lower fitness value than its opposite guess. Therefore, starting with the fitter of the two guesses has the potential to accelerate convergence. The same approach can be applied not only to initial solutions but also continuously to each solution in the current population. Also, when the population begins to converge into a smaller neighborhood surrounding an optimum, taking opposition moves can increase diversity of the population. In addition, when the population converges, the magnitude of difference vectors will become smaller. However, difference vectors generated by using parents that just underwent an opposite move will be large thereby resulting in larger perturbation in the mutant vector. Therefore, ODE possesses superior capability to jump out of local optima basins.

Before discussing the steps of ODE, below we define o*pposite numbers*.

*Definition 1:* Let $x$ be a real number defined in the closed interval $[a, b]$, i.e., $x \in [a, b]$. Then the opposite number $\overset{\cup}{x}$ of $x$ may be defined as

$$\overset{\cup}{x} = a + b - x. \quad (26)$$

The ODE changes the classical DE using the concept of opposite numbers at the following three different stages.

1) *Opposition based population initialization*: first a uniformly distributed random population $P(NP)$ is generated and then the opposite population $OP(NP)$ is calculated. The $k$th opposite individual corresponding to the $k$th parameter vector of $P(NP)$ is [following (26)], $OP_{k,j} = a_{k,j} + b_{k,j} - P_{k,j}$, where $k = 1, 2, ...., NP$ and $j = 1, 2, ...., D$, $a_{k,j}$ and $b_{k,j}$ denote the interval boundaries of $j$-th parameter of the $k$th vector, i.e., $x_{k,j} \in [a_{k,j}, b_{k,j}]$. Finally, $NP$ fittest individuals are selected from the set $\{P(NP), OP(NP)\}$ as the initial population.
2) *Opposition based generation jumping*: in this stage, after each iteration, instead of generating new population by evolutionary process, the opposite population is calculated with a predetermined probability $Jr(\in (0, 0.04))$ and the $NP$ fittest individuals may be selected from the current population and the corresponding opposite population.
3) *Opposition based best individual jumping*: in this phase, at first a difference-offspring of the best individual in the current population is created as

$$\vec{X}_{new\_best,G} = \vec{X}_{best,G} + F' \cdot (\vec{X}_{r_1,G} - \vec{X}_{r_2,G})$$

where $r_1$ and $r_2$ are mutually different random integer indices selected from $\{1, 2, ..., NP\}$ and $F'$

is a real constant. Next the opposite of offspring is generated as $\vec{X}_{opp\_newbest,G}$. Finally, the current best member is replaced by the fittest member of the set $\{\vec{X}_{best,G}, \vec{X}_{new\_best,G}, \vec{X}_{opp\_newbest,G}$.

### E. DE with Neighborhood-Based Mutation

The efficiency of most EAs depends on their extent of explorative and exploitative tendencies during the course of search. Exploitation means the ability of a search algorithm to use the information already collected and thus to orient the search more toward the goal while exploration is the process that allows introduction of new information into the population. Exploration helps the algorithm to quickly search the new regions of a large search-volume. Das *et al.* [21] proposed two kinds of topological neighborhood models for DE in order to achieve better balance between its explorative and exploitative tendencies. The resulting algorithm, called DEGL, was put forward as an improvement over the DE/target-to-best/1 scheme, which shows a poor performance on multimodal fitness landscapes, as noted in the studies of Mezura-Montes *et al.* [56] and Price *et al.*. [74, p. 156].

Suppose we have a DE population $P_G = [\vec{X}_{1,G}, \vec{X}_{2,G}, ...., \vec{X}_{NP,G}]$ at generation $G$. The vector indices are sorted only randomly (as obtained during initialization) in order to preserve the diversity of each neighborhood. Now for every vector $\vec{X}_{i,G}$ we define a neighborhood of radius $k$ (where $k$ is a non-zero integer from 0 to $(NP-1)/2$ as the neighborhood size must be smaller than the population size, i.e., $2k+1 \leq NP$), consisting of vectors $\vec{X}_{i-k,G}, ..., \vec{X}_{i,G}, ..., \vec{X}_{i+k,G}$. We assume the vectors to be organized on a ring topology with respect to their indices, such that vectors $\vec{X}_{NP,G}$ and $\vec{X}_{2,G}$ are the two immediate neighbors of vector $\vec{X}_{1,G}$. For each member of the population a local donor vector is created by employing the best (fittest) vector in the neighborhood of that member and any two other vectors chosen from the same neighborhood. The model may be expressed as

$$\vec{L}_{i,G} = \vec{X}_{i,G} + \alpha \cdot (\vec{X}_{n\_best_i,G} - \vec{X}_{i,G}) + \beta \cdot (\vec{X}_{p,G} - \vec{X}_{q,G}) \quad (27)$$

where the subscript $n\_best_i$ indicates the best vector in the neighborhood of $\vec{X}_{i,G}$ and $p, q \in [i-k, i+k]$ with $p \neq q \neq i$. Similarly, the global donor vector is created as

$$\vec{g}_{i,G} = \vec{X}_{i,G} + \alpha \cdot (\vec{X}_{g\_best,G} - \vec{X}_{i,G}) + \beta \cdot (\vec{X}_{r_1,G} - \vec{X}_{r_2,G}) \quad (28)$$

where the subscript $g\_best$ indicates the best vector in the entire population at iteration $G$ and $r_1, r_2 \in [1, NP]$ with $r_1 \neq r_2 \neq i$. $\alpha$ and $\beta$ are the scaling factors. Now we combine the local and global donor vectors using a scalar weight $w \in (0, 1)$ to form the actual donor vector of the proposed algorithm

$$\vec{V}_{i,G} = w \cdot \vec{g}_{i,G} + (1-w) \cdot \vec{L}_{i,G}. \quad (29)$$

Clearly, if $w = 1$ and in addition $\alpha = \beta = F$, the donor vector generation scheme in (30) reduces to that of DE/target-to-best/1. Hence, the latter may be considered as a special case of this more general strategy involving both global and local neighborhood of each vector synergistically.

Note that DE/target-to-best/1, in its present form, favors exploitation only, since all the vectors are attracted toward the same best position found so far by the entire population, thereby converging faster toward the same point. In DEGL, a vector's neighborhood is the set of other parameter vectors that it is connected to; it considers their experience when updating its position. The graph of inter-connections is called the neighborhood structure. Generally, neighborhood connections are independent of the positions pointed to by the vectors. In the *local* model, whenever a parameter vector points to a good region of the search space, it only directly influences its immediate neighbors. Its second degree neighbors will only be influenced after those directly connected to them become highly successful themselves. Thus, there is a delay in the information spread through the population regarding the best position of each neighborhood. Therefore, the attraction to specific points is weaker, which reduces the chances of getting trapped in local minima.

### F. DE with Adaptive Selection of Mutation Strategies

DE can encompass a number of trial vector generation strategies, each of which may be effective over certain problems but poorly perform over the others [26]. In [76], for the first time Qin *et al*. proposed a self-adaptive variant of DE (SaDE), where along with the control parameter values the trial vector generation strategies are also gradually self-adapted by learning from their previous experiences in generating promising solutions. Consequently, it is possible to determine a more suitable generation strategy along with its parameter settings adaptively to match different phases of the search process/evolution.

In SaDE, four effective trial vector generation strategies namely the DE/rand/1/bin, DE/rand-to-best/2/bin, DE/rand/2/bin and finally DE/current-to-rand/1 were chosen to constitute a strategy candidate pool. The first three DE-variants are equipped with binomial type crossover while the last one uses arithmetic recombination (described in Section IV-B). In the SaDE algorithm, for each target vector in the current population, one trial vector generation strategy is selected from the candidate pool according to the probability learned from its success rate in generating improved solutions (that can survive to the next generation) within a certain number of previous generations, called the learning period (LP). The selected strategy is subsequently applied to the corresponding target vector to generate a trial vector. More specifically, at each generation, the probabilities of choosing each strategy in the candidate pool are summed to 1. These probabilities are initially equal ($1/K$ for $K$ total number of strategies in the pool) and are then gradually adapted during evolution, based on the *Success* and *Failure Rates* [76] over the previous *LP* generations. The adaptations of the probabilities take place in such a fashion that, the larger the success rate for the $k$th strategy in the pool within the previous *LP* generations, the larger is the probability of applying it to generate trial vectors at the current generation.

The performance of SaDE was compared with the conventional DE and three adaptive DE-variants: ADE [S29], SDE [65], and jDE [10] (already discussed in Section III) over a

suite of 26 bound constrained numerical optimization problems, and the authors reported that, SaDE was more effective in obtaining better quality solutions, with the relatively smaller standard deviations, and higher success rates.

### G. Adaptive DE with DE/Current-to-pbest Mutation

In order to avoid the need for problem specific parameter tuning and also to improve the convergence characteristics of DE, an adaptive DE-variant, called JADE, was recently proposed [118]. The algorithm implements a new mutation strategy, referred by the authors as DE/current-to-*p*best and uses an optional external archive to track the previous history of success and failure. It also updates the control parameters in an adaptive manner with generations. The DE/current-to-*p*best strategy is a less greedy generalization of the DE/current-to-best/ strategy. Instead of only adopting the best individual in the DE/current-to-best/1 strategy, the current-to-pbest/1 strategy utilizes the information of other good solutions. Moreover, the recently explored inferior solutions are also incorporated in this strategy. The DE/current-to-pbest/1 without external archive generates the donor vector as

$$\vec{V}_{i,G} = \vec{X}_{i,G} + F_i \cdot (\vec{X}^p_{best,G} - \vec{X}_{i,G}) + F_i \cdot (\vec{X}_{r^i_1,G} - \vec{X}_{r^i_2,G}) \quad (30)$$

where $\vec{X}^p_{best,G}$ is randomly chosen as one of the top 100p% individuals of the current population with $p \in (0, 1]$. $F_i$ is the scale factor associated with the $i$th individual and it is updated dynamically in each generation. JADE can optionally make use of an external archive, which stores the recently explored inferior solutions. Let $A$ denote the archive of inferior solutions and $P$ denote the current population. Then DE/current-to-pbest/1 with external archive generates the donor vector as

$$\vec{V}_{i,G} = \vec{X}_{i,G} + F_i \cdot (\vec{X}^p_{best,G} - \vec{X}_{i,G}) + F_i \cdot (\vec{X}_{r^i_1,G} - \vec{X}'_{r^i_2,G}) \quad (31)$$

where $\vec{X}_{i,G}$, $\vec{X}^p_{best,G}$, and $\vec{X}_{r^i_1,G}$ are selected from $P$ as before in (30), but $\vec{X}'_{r^i_2,G}$ is selected at random from the union $P \bigcup A$, of the current population and archive. The archive operation is made very simple to avoid significant computation overhead. Initially the archive is empty. Then, after each generation, the parent solutions that fail in the selection process are added to the archive. If the archive size exceeds a certain threshold, then some solutions are randomly eliminated from the archive to keep the archive size fixed.

### H. Hybrid DE Algorithms

Hybridisation, in context to metaheuristics, primarily refers to the process of combining the best features of two or more algorithms together, to form a new algorithm that is expected to outperform its ancestors over application-specific or general benchmark problems. Over the past few years, DE has been successfully hybridized with several other global optimization algorithms like PSO [S38], ant colony systems [S39], artificial immune systems (AIS) [S40], bacterial foraging optimization algorithm (BFOA) [S41], and simulated annealing (SA) [S42]. Also researchers attempted to embed different local search techniques in basic DE, to improve its exploitation abilities. In this section, we shall discuss the hybrid DE algorithms in two parts: first one will present the synergy between DE and other global search methods while the second one will review the blending of DE with local search algorithms.

1) *Synergy of DE with Other Global Optimization Algorithms:* The concept of particle swarms, although initially introduced for simulating human social behaviors, has become very popular these days as an efficient global search and optimization technique. The first synergy between DE and PSO was reported by Hendtlass, who proposed a combined swarm differential evolution algorithm [S43] serving as a hybrid optimizer based on PSO and DE. In this optimizer, particle positions are updated only if their offspring have better fitness. DE acts on the particles in the PSO swarm at specified intervals. Zang and Xie [119] proposed another popular hybrid algorithm called DEPSO, in which the original PSO algorithm and the DE operator alternate at the odd iterations and at the even iterations. DEPSO achieved better convergence results than both the original algorithms over certain constrained optimization problems. Das *et al.* [16] presented a tightly coupled synergy of PSO and DE, called particle swarm optimization with differentially perturbed velocity (PSO-DV). PSO-DV introduces a differential operator (borrowed from DE) in the velocity-update scheme of PSO. Further, unlike conventional PSO, a particle is actually shifted to a new location only if the new location yields a better fitness value, i.e., a DE-type selection strategy has been incorporated into the swarm dynamics.

In [58], Moore and Venayagamoorthy proposed a new hybrid of DE and PSO, which is similar in spirit to the algorithm proposed in [119], but the DE and PSO in it are DE/*rand*/*2*/*bin* and a modified PSO with "*Ring*" topology respectively. In [S44], Liu *et al*. proposed a similar DEPSO and used it to train artificial neural networks. Like the work reported in [119], the PSO in this hybrid optimizer is also based on *Gbest* model; however, the DE in it is DE/*target-to-best*/*1*/*bin*. In particular, this hybrid also adopts a chaotic local search to improve its local exploitation ability. In 2004, Kannan *et al*. [S45] proposed a distinctive DEPSO (named C-PSO in [S45]). The DE algorithm in it is employed to select three control parameters on-line for PSO. In other words, DE serves as a meta-optimizer for the optimization of PSOs search behavior. Recently Hao *et al*. [S46] constructed a new hybrid optimizer, where DE and PSO are regarded as two operators to generate candidate solutions, and they act on the level of dimensional components of individuals.

In [66], Omran *et al*. presented two hybrids of DE and PSO. The first DEPSO (named DEPSO-OES) is somewhat similar to the hybrid described in [S46]. The DE (DE/*rand*/*1*/*bin*) and PSO-cf (PSO with constriction factor) in it also alternate in a stochastic way, but both DE and PSO act on the level of a whole individual, that is to say, each individual at each generation has only one updating method (DE or PSO). Besides, the probability for controlling the selection of updating method and the scaling factor in DE are dynamic and adaptive. The second hybrid method combined the bare bones PSO proposed by Kennedy [S47] and DE in an embedding way. Xue *et al.* described another scheme of mixing DE operators with PSO in [S48].

Das *et al.* [19] modified the selection mechanism of the classical DE family by using the concepts of SA such that the probability of accepting the inferior solutions may be dynamically altered with iterations. Biswas *et al.* [9] proposed a synergistic coupling of DE and BFOA. Foraging can be modeled as an optimization process where an animal seeks to maximize energy per unit time spent for foraging. BFOA emulates the foraging behavior of a group of *Escherichia coli* bacteria living in our intestine. In [9], the computational chemotaxis step of BFOA, which may also be viewed as a stochastic gradient search, has been coupled with DE type mutation and crossing over of the optimization agents leading to the new hybrid algorithm called chemotactic differential evolution (CDE). In [S49] and [S50], hybridizations of the DE with an ant colony optimizer are proposed. In [S51], He and Han propose a hybrid binary DE based on AIS for tackling discrete optimization problems. Kaelo and Ali [33] use the attraction-repulsion concept of electromagnetism-like algorithm to boost the mutation operation of the original DE.

2) *Synergy of DE with Local Search Methods:* Local search algorithms primarily explore a small neighborhood of a candidate solution in the search space until a locally optimal point is found or a time bound is elapsed. Noman and Iba [64] proposed a crossover based adaptive local search (LS) operation to improve the performance of the classical DE. Typically in LS, every candidate solution has more than one neighbour solution; the choice of which one to move to is taken using only information about the solutions in the neighbourhood of the current one, hence the name local search. If the choice of the neighbouring solution is done by taking the one locally maximizing the criterion, the metaheuristic takes the name *hill-climbing*. The authors in [64] proposed an LS, whose length of the search can be adjusted adaptively using a hill-climbing heuristic. The incorporation of a crossover-based local search (XLS) with adaptive length (adaptive length XLS, shortened as AHCXLS) in DE resulted into a DE-variant called by the authors: DEahcSPX, where SPX is the simplex-based crossover scheme proposed by Tsutsui *et al.* for real-coded GAs [S52].

The experimental results reported by Noman and Iba [64] indicated that DEahcSPX could outperform the classical DE (DE/rand/1/bin) in terms of convergence speed over a set of carefully chosen numerical benchmarks [95]. The overall performance of the adaptive LS scheme was reportedly better than the other crossover-based LS strategies and the overall performance of the newly proposed DE algorithm was shown to be superior to or at least comparable with some other memetic algorithms (MAs) [S53] selected from literature.

Yang *et al.* [108] proposed a hybridization of DE with the neighborhood search, which appears as a main strategy underpinning EP [S54]. The resulting algorithm, known as NSDE, performs mutation by adding a normally distributed random value to each target-vector component in the following way:

$$\vec{V}_{i,G} = \vec{X}_{r_1^i,G} + \begin{cases} \vec{d}_{i,G} \cdot N(0.5, 0.5) & \text{if } rand_i(0,1) < 0.5 \\ \vec{d}_{i,G} \cdot \delta & \text{otherwise} \end{cases} \tag{32}$$

where $\vec{d}_{i,G} = \vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}$ is the usual difference vector and $\delta$ denotes a Cauchy random variable with scale parameter $t = 1$. Recently Yang *et al.* [110] used a Self-adaptive NSDE in the cooperative coevolution framework that is capable of optimizing large-scale non-separable problems (up to 1000 dimensions). They proposed a random grouping scheme and adaptive weighting for problem decomposition and coevolution. Somewhat similar in spirit to this paper is the study by Yang *et al.* [S55] on self-adaptive DE with neighborhood search (SaNSDE). SaNSDE incorporates self-adaptation ideas from the Qin *et al*'s SaDE [76] and proposes three self-adaptive strategies: self-adaptive choice of the mutation strategy between two alternatives, self-adaptation of the scale factor $F$, and self-adaptation of the crossover rate $Cr$. In contrast to Yang *et al*.'s works on NSDE and SaNSDE, in the topological neighborhood-based mutation scheme proposed in [21], the authors keep the scale factor non-random and use a ring-shaped neighborhood topology (inspired by PSO [S56]), defined on the index graph of the parameter vectors, to derive a local neighborhood-based mutation model. Also instead of $F$ and $Cr$, the weight factor that unifies two kinds of mutation models, have been made self-adaptive in one of the variants of the algorithms described in [21].

MAs represent one of the recent growing areas of research in evolutionary computation. The term MA is now widely used to denote a synergy of evolutionary or any population-based approach with separate individual learning or local improvement procedures for problem search. Neri and Tirronen [59] proposed a DE-based MA, which employs within a self-adaptive scheme, two local search algorithms. The algorithm was referred by authors as the scale factor local search differential evolution. These local search algorithms aim at detecting a value of the scale factor $F$ corresponding to an offspring with a higher fitness, while the generation is executed. The local search algorithms thus assist in the global search and generate offspring with a higher fitness, which are subsequently supposed to promote the generation of enhanced solutions within the evolutionary framework. In [S57], Tirronen *et al*. proposed a DE-based MA employing three local search algorithms coordinated by means of fitness diversity adaptation and a probabilistic scheme for designing digital filters, which aim at detecting defects of the paper produced during an industrial process. In [14], Caponio *et al*. incorporated PSO and two other LS algorithms (Nelder mead algorithm and Rosenbrock algorithm) in the framework of DE. The main idea is that initially PSO should quickly improve a solution having poor fitness and include it in the DE population. This solution (called by the authors as "*super-fit individual*") should therefore be the one leading the DE-search. The two local searchers are invoked within the main DE-search probabilistically. Although the paper reports improvement of the gross performance of DE, role of the LS algorithms are not much clear.

3) *DE-Variants for Discrete and Binary Optimization:* Although DE was devised mainly for real parameter optimization, over the years researchers have tried to modify it for tackling binary and discrete optimization problems as well. In the early days of DE research, Lampinen and Zelinka first focused in this direction through their conference article in MENDEL'99 [40]. For handling of integer variables, they

recommended truncating the parameter values for objective function evaluation such that the population of DE still works with floating-point values. They pointed out that although such truncation changes the effective objective function landscape from DE's point of view by introducing flat areas to the fitness landscape, DE's self-adaptive reproduction scheme is well able to move across to those flat areas. In the same paper, Lampinen and Zelinka also came up with a straightforward approach for optimizing discrete parameters that are limited to a set of standard values. For example, the thickness of a steel plate, the diameter of a copper pipe, the size of a screw, the size of a roller bearing, and so on, are often limited to a set of commercially available standard sizes. A discrete value is optimized indirectly so that DE actually works on an integer value (index) that points to the actual discrete value. First, the discrete set of available values is arranged to an ascending sequence, and then an index is assigned to refer each available value. DE works with these indices by optimizing the index like any integer variable. However, for objective function evaluation the actual discrete value pointed by the index is used.

In [S212], Tasgetiren *et al*. presented a DE algorithm to solve the permutation flowshop scheduling problem with the makespan criterion. DE was a traditional continuous algorithm and the smallest position value rule was presented to convert the continuous vector to a discrete job permutation. In [S213], Onwubolu and Davendra presented a DE variant for solving scheduling problems. In [S58], Tasgetiren *et al*. proposed a discrete differential evolution algorithm (DDE) for the no-wait flowshop scheduling problem with total flow time criterion. In the DDE they proposed, a discrete version of DE based on a insert mutation and PTL crossover operator they offered are employed. In order to further improve the solution quality, a variable neighborhood descent local search is embedded in the DDE algorithm. A DDE algorithm was presented by Tasgetiren *et al.* [S214] for the total earliness and tardiness penalties with a common due date on a single-machine. In [S214], the same mutation and the PTL crossover operator were used in the binary context as well as a Bswap local search is employed to further improve the solution quality. A similar approach but working on a continuous domain was presented in Nearchou [S215] to solve the total earliness and tardiness penalties with a common due date on a single-machine. In [S215], the conversion of continuous vector was based on the fact that a value less than or equal to 0.5 in the string indicates that the corresponding job is early, otherwise the job is late. In [S216], Al-Anzi and Allahverdi proposed a self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times. Later, Pan *et al.* [217] presented a DDE based on the one in [S58] to solve the permutation flowshop scheduling problem. Furthermore, Qian *et al.* [S218] proposed another DE-based approach to solve the no-wait flowshop scheduling problem. Tasgetiren *et al.* [S59] developed a DDE for the single machine total weighted tardiness problem with sequence dependent setup times where novel speed-up methods were presented. In [S219], Pan *et al.* developed a novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems. Wang *et al.* [220] proposed a hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. Another bi-criteria DE was presented by Qian *et al.* in [S221] to solve the multiobjective flow shop scheduling with limited buffers. In [S222], Tasgetiren *et al.* proposed an ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem. The novelty in [S222] stems from the fact that the ensemble of destruction and construction procedures of iterated greedy algorithm and crossover operators are achieved in parallel populations. In addition, Damak *et al.* presented [S223] a DE variant for solving multimode resource-constrained project scheduling problems. In [S224] and [S225], DDE was applied to solve the no-idle permutation flow shop scheduling problems. Additional discrete and combinatorial applications of DE algorithms were presented in detail in [S226] and [S227].

Recently, Pampara *et al.* [67] proposed a new DE variant that can operate in binary problem spaces without deviating from the basic search mechanism of the classical DE. The algorithm was named by its authors as the angle modulated DE as it employs a trigonometric function as a bit string generator. The trigonometric generating function used in the angle modulation function is a composite sinusoidal function, which may be given as

$$g(x) = \sin(2\pi(x-a) \times b \times \cos A)) + d \tag{33}$$

where $A = 2\pi \times c \times (x-a)$ and $x$ is a single element from a set of evenly separated intervals determined by the required number of bits that need to be generated. The DE is used to evolve the coefficients to the trigonometric function ($a$, $b$, $c$, $d$), thereby allowing a mapping from continuous-space to binary-space. Instead of evolving the higher-dimensional binary solution directly, angle modulation is used together with DE to reduce the complexity of the problem into a 4-D continuous-valued problem. Yuan *et al*. [S60] used a discrete binary differential evolution approach to solve the unit commitment problem.

### *I. Parallel DE*

Exploiting the huge development of computational resources (both software and hardware), parallel computing has emerged as a form of high-performance computation, where many calculations are carried out simultaneously, based on the principle that large problems can often be divided into smaller ones, which are then solved concurrently (in parallel). Like other EAs, DE can also be parallelized (mainly for improving its speed and accuracy on expensive optimization problems) owing to the fact that each member of the population is evaluated independently. The only phase in the algorithm that necessitates communication with other individuals is reproduction. This phase can also be made parallel for pair of vectors.

The first attempt to distribute DE across a cluster of computers (connected through local area networks) was made by Lampinen [38]. In his method, the whole population is kept in a master processor that selects individuals for mating and sends them to slave processors for performing other operations. Lampinen's parallelization scheme could also overcome

the drawbacks due to the heterogeneous speed of the slave processors. Tasoulis *et al*. [101] proposed a parallel DE scheme that maps an entire subpopulation to a processor, allowing different subpopulations to evolve independently toward a solution. To promote information sharing, the best individual of each subpopulation is allowed to move to other subpopulations according to a predefined topology. This operation is known as *migration* in parallel EA literature [S194] and island model GAs.

During migration, instead of replacing a randomly chosen individual from a subpopulation, Kozlov and Samsonov [S195] suggested to replace the oldest member by the best member of another subpopulation in the topological neighborhood of the former subpopulation. Following the work of [101], Weber *et al*. [S196] proposed a scale factor ($F$) inheritance mechanism in conjunction with distributed DE with ring topology based migration scheme. In this framework, each sub-population is characterized by its own scale factor value. With a probabilistic criterion, that individual displaying the best performance is migrated to the neighbor population and replaces a randomly selected individual of the target subpopulation. The target sub-population inherits not only this individual but also the scale factor if it seems promising at the current stage of evolution.

## V. DE in Complex Environments

This section reviews the extensions of DE for handling multiobjective, constrained, and large scale optimization problems. It also surveys the modifications of DE for optimization in dynamic and uncertain environments.

### A. DE for Multiobjective Optimization

Due to the multiple criteria nature of most real-world problems, multiobjective optimization (MO) problems are ubiquitous, particularly throughout engineering applications. As the name indicates, multiobjective optimization problems involve multiple objectives, which should be optimized simultaneously and that often are in conflict with each other. This results in a group of alternative solutions, which must be considered equivalent in the absence of information concerning the relevance of the others. The concepts of *dominance* and *Pareto-optimality* may be presented more formally in the following way.

*Definition 2:* Consider without loss of generality the following multiobjective optimization problem with $D$ decision variables $x$ (parameters) and $n$ objectives $y$:

$$\text{Minimize } \vec{Y} = f(\vec{X}) = (f_1(x_1, ...., x_D), ...., f_n(x_1, ...., x_D)) \tag{34}$$

where $\vec{X} = [x_1, ....., x_D]^T \in P$ and $\vec{Y} = [y_1, ...., y_n]^T \in O$ and where $\vec{X}$ is called decision (parameter) vector, $P$ is the parameter space, $\vec{Y}$ is the objective vector, and $O$ is the objective space. A decision vector $\vec{A} \in P$ is said to dominate another decision vector $\vec{B} \in P$ (also written as $\vec{A} \prec \vec{B}$ for minimization) if and only if

$$\begin{aligned} &\forall i \in \{1, ...., n\} : && f_i(\vec{A}) \leq f_i(\vec{B}) \\ \wedge \quad &\exists j \in \{1, ....., n\} : && f_j(\vec{A}) < f_j(\vec{B}). \end{aligned} \tag{35}$$

Based on this convention, we can define non-dominated, *Pareto-optimal* solutions as follows.

*Definition 3:* Let $\vec{A} \in P$ be an arbitrary decision vector.

1) The decision vector $\vec{A}$ *is said to be* non-dominated regarding the set $P' \subseteq P$ if and only if there is no vector in $P'$ which can dominate $\vec{A}$.
2) The decision (parameter) vector $\vec{A}$ is called Pareto-optimal if and only if $\vec{A}$ is non-dominated regarding the whole parameter space $P$.

Many evolutionary algorithms were formulated by the researchers to tackle multiobjective problems in recent past [S61], [S62]. Apparently, the first paper that extends DE for handling MO problems is by Chang *et al*. [S63] and it bases itself on the idea of Pareto dominance. DE/rand/1/bin with an external archive (called "Pareto optimal set" by the authors and also known as the current non-dominated set) is used to store the non-dominated solutions obtained during the search. The approach also incorporates fitness sharing to maintain diversity. Abbas and Sarkar presented the Pareto differential evolution (PDE) algorithm [2] for MO problems with continuous variables and achieved very competitive results compared to other evolution algorithms in MO literature. However, there is no obvious way to select best crossover and mutation rates apart from running the algorithm with different rates. It handles only one (main) population. Reproduction is undertaken only among non-dominated solutions, and offspring are placed into the population if they dominate the main parent. A distance metric relationship is used to maintain diversity. In [S64], Abbass presented an approach called Memetic Pareto artificial neural networks. This approach consists of PDE enhanced with the back-propagation local search algorithm, in order to speed up convergence.

Kukkonen and Lampinen extended DE/rand/1/bin to solve multiobjective optimization problems in their approach called generalized differential evolution (GDE). In the first version of their approach [S65], the authors modified the original DE selection operation by introducing Pareto dominance as a selection criterion while in a second version, called GDE2 [S66] a crowding distance measure was used to select the best solution. To deal with the shortcomings of GDE2 regarding slow convergence, Kukkonen and Lampinen proposed an improved version called GDE3 [35] (a combination of the earlier GDE versions and the Pareto-based differential evolution algorithm [S67]). This version added a growing population size and nondominated sorting (as in the NSGA-II [S68]) to improve the distribution of solutions in the final Pareto front and to decrease the sensitivity of the approach to its initial parameters. Santana-Quintero and Coello Coello proposed the $\in$-MyDE in [86]. This approach keeps two populations: the main population (which is used to select the parents) and a secondary (external) population, in which the concept of $\in$-dominance [S69] is adopted to retain the non-dominated solutions found and to distribute them in a uniform way.

In [105], Xue *et al.* came up with the multiobjective DE (MODE) in which the best individual is adopted to create the offspring. A Pareto-based approach is introduced to implement

the selection of the best individual. If a solution is dominated, a set of non-dominated individuals can be identified and the "best" turns out to be any individual (randomly picked) from this set. Also, the authors adopt $(\mu+\lambda)$ selection, Pareto ranking and crowding distance in order to produce and maintain well-distributed solutions. Robic and Filipic presented a DE for multiobjective optimization (called DEMO) in [83]. This algorithm combines the advantages of DE with the mechanisms of Pareto-based ranking and crowding distance sorting. DEMO only maintains one population and it is extended when newly created candidates take part immediately in the creation of the subsequent candidates. This enables a fast convergence toward the true Pareto front, while the use of non-dominated sorting and crowding distance (derived from the NSGA-II [S68]) of the extended population promotes the uniform spread of solutions. Iorio and Li [32] proposed the non-dominated sorting DE (NSDE), which is a simple modification of the NSGA-II [S68]. The only difference between this approach and the NSGA-II is in the method for generating new individuals. The NSGA-II uses a real-coded crossover and mutation operator, but in the NSDE, these operators were replaced with the operators of differential evolution. NSDE was shown to outperform NSGA-II on set of rotated MO problems with strong interdependence of variables.

Some researchers have proposed approaches that use non-Pareto based multiobjective concepts like combination of functions, problem transformation, and so on. For example, Babu and Jehan [6] proposed a DE algorithm for MO problems, which uses the DE/rand/1/bin variant with two different mechanisms to solve bi-objective problems: first, incorporating one objective function as a constraint, and secondly using an aggregating function**.** Li and Zhang [S70], [46] proposed a multiobjective differential evolution algorithm based on decomposition (MOEA/D-DE) for continuous multiobjective optimization problems with variable linkages. The DE/rand/1/bin scheme is used for generating new trial solutions, and a neighborhood relationship among all the sub-problems generated is defined, such that they all have similar optimal solutions. In [46], they introduce a general class of continuous MO problems with complicated Pareto set (PS) shapes and reported the superiority of MOEA/D-DE over NSGA-II with DE type reproduction operators. Summation of normalized objective values with diversified selection approach was used in [79] without the need for performing non-dominated sorting.

Some authors also consider approaches where a set of schemes have been mixed in the DE-based multiobjective algorithm. Examples of such combined techniques are the vector evaluated DE [70] by Parsopoulos *et al* and the work of Landa-Becerra and Coello Coello [42] where they hybridized the $\varepsilon$-constraint technique [S71] with a single-objective evolutionary optimizer: the cultured DE [43]. Recently the concept of self-adaptive DE has been extended to handle MO problems in [29], [30], and [116].

### *B. DE for Constrained Optimization*

Most of the real world optimization problems involve finding a solution that not only is optimal, but also satisfies one or more constraints. A general formulation for constrained optimization may be given in the following way.

*Definition 4:* Find $\vec{X} = [x_1, x_2, ..., x_D]^T \qquad \vec{X} \in \Re^D$

$$\text{to minimize: } f(\vec{X}) \tag{36a}$$

subjected to

$$\text{inequality constraints: } g_i(\vec{X}) \leq 0 \qquad i = 1, 2, ..., K \tag{36b}$$

$$\text{equality constraints: } h_j(\vec{X}) = 0 \qquad j = 1, 2, ..., N \tag{36c}$$

$$\text{and boundary constraints: } x_{j,min} \leq x_j \leq x_{j,\max}. \tag{36d}$$

Boundary constraints are very common in real-world applications, often because parameters are related to physical components or measures that have natural bounds, e.g., the resistance of a wire or the mass of an object can never be negative. In order to tackle boundary constraints, penalty methods drive solutions from restricted areas through the action of an objective function-based criterion. DE uses the following four kinds of penalty method to handle boundary constraint violation.

1) *Brick wall penalty* [74]*:* if any parameter of a vector falls beyond the pre-defined lower or upper bounds, objective function value of the vector is made high enough (by a fixed big number) to guarantee that it never gets selected.
2) *Adaptive penalty* [90], [91]*:* similar to brick wall penalty, but here the increase in the objective function value of the offender vector may depend on the number of parameters violating bound constraints and their magnitudes of violation.
3) *Random reinitialization* [40], [74]*:* replaces a parameter that exceeds its bounds by a randomly chosen value from within the allowed range following (1).
4) *Bounce-back* [74]*:* relocates the parameter in between the bound it exceeded and the corresponding parameter from the base vector.

The first known extension of DE toward the handling of inequality constrained optimization problems (mainly design centering) was by R. Storn [93]. He proposed a multimember DE (called CADE: constraint adaptation with DE, in his paper) that generates $M$ $(M > 1)$ children for each individual with three randomly selected distinct individuals in the current generation, and then only one of the $M + 1$ individuals will survive in the next generation. Mezura-Montes *et al*. [S72] used the concept also to solve constrained optimization problems. Zhang *et al*. [117] mixed the dynamic stochastic ranking with the multimember DE framework and obtained promising performance on the 22 benchmarks taken from the CEC 2006 competition on constrained optimization [47].

Lampinen applied DE to tackle constrained problems [39] by using Pareto dominance in the constraints space. Mezura-Montes *et al*. [S73] proposed to add Deb's feasibility rules [S74] into DE to deal with constraints. Kukkonen and Lampinen [36] presented a generalised DE-based approach to solve constrained multiobjective optimization problems.

Zielinsky and Laur [121] also used Deb's rules [S74] with DE to solve some constrained optimization problems. Some researchers have also tried hybrid approaches such as the DE with gradient-based mutation (where gradients were derived from the constraint equations) by Takahama and Sakai [97] and PSO-DE (PESO+) by Muñoz-Zavala *et al*. [S75]. The $\varepsilon$-DE algorithm of Takahama and Sakai [97] uses a dynamic control of the allowable constraint violation specified by the $\varepsilon$-level and it achieved the first rank in the CEC 2006 competition on the constrained real-parameter optimization [47]. Tasgetiren and Suganthan presented a multi-populated DE algorithm [99] to solve real-parameter constrained optimization problems. They employed the notion of a near feasibility threshold in the proposed algorithm to penalize infeasible solutions. Mezura-Montes *et al*. [57] proposed a DE approach that attempts to increase the probability of each parent to generate a better offspring. This is done by allowing each solution to generate more than one offspring but using a different mutation operator, which combines information of the best solution in the population and also information of the current parent to find new search directions.

On the other hand, some studies have also been reported regarding parameter control in DE for constrained optimization. Brest *et al.* [S76] have proposed an adaptive parameter control for two DE parameters related to the crossover and mutation operators. Huang *et al*. [28] used an adaptive mechanism to select among a set of DE variants to be used for the generation of new vectors based on a success measure. Moreover, they also adapted some DE parameters to control the variation operators. Very recently Mezura-Montes and Palomeque-Ortiz [S77] presented the adaptive parameter control in the diversity differential evolution (DDE) [S72] algorithm for constrained optimization. Three parameters namely the scale factor $F$, the crossover rate $Cr$, and the number of offspring generated by each target vector $NO$, are self-adapted by encoding them within each individual and a fourth parameter called selection ratio $S_r$ is controlled by a deterministic approach.

Huang *et al*. [31] presented a cooperative-coevolutionary approach in conjunction with DE for constrained optimization problems. In their algorithm first, a special penalty function is designed to handle the constraints. Second, a co-evolution model is presented and DE is employed to perform evolutionary search in spaces of both solutions and penalty factors. Thus, the solutions and penalty factors evolve interactively and self-adaptively, and both the satisfactory solutions and suitable penalty factors can be obtained simultaneously. Recently Ali and Kajee-Bagdadi proposed a local exploration-based DE [4] for constrained global optimization. They used a restricted version of the pattern search (PS) method [S78] as their local technique. Constraint handling methods such as the superiority of feasible points and the parameter free penalty are also employed. Recently Santana-Quintero *et al*. [87] extended the PDE [1], [2] to handle constrained MO problems by using a two-stage hybrid DE approach where in the first one, an MO version of DE is used to generate an initial approximation of the Pareto front. Then, in the second stage, rough set theory is used to improve the spread and quality of this initial approximation.

### C. DE for Large-Scale Optimization

In the past two decades, several kinds of nature-inspired optimization algorithms have been designed and applied to solve optimization problems. Although these approaches have shown excellent search abilities when applied to some 30–100 dimensional problems, usually their performance deteriorates quickly as the dimensionality of search space increases beyond 500. The reasons appear to be two-fold. First, complexity of the problem usually increases with the size of problem, and a previously successful search strategy may no longer be capable of finding the optimal solution. Second, the solution space of the problem increases exponentially with the problem size, and a more efficient search strategy is required to explore all the promising regions in a given time budget. Since the performance of basic DE schemes also degrade with massive increase in problem dimensions, some important attempts have been made by the researchers to make DE suitable for handling such large-scale optimization problems.

In [62], Noman and Iba proposed fittest individual refinement (FIR), a crossover based local search method for DE, such that the FIR scheme accelerates DE by enhancing its search capability through exploration of the neighborhood of the best solution in successive generations. The proposed memetic version of DE (augmented by FIR) was shown to obtain an acceptable solution with a lower number of evaluations particularly for higher dimensional functions. Another memetic DE for high-dimensional optimization was presented by Gao and Wang [27], where the stochastic properties of chaotic system is used to spread the individuals in search spaces as much as possible and the simplex search method is employed to speed up the local exploiting and the DE operators help the algorithm to jump to a better point.

In terms of optimizing high-dimensional problems, cooperative co-evolution (first proposed by Potter and De Jong for GAs [S79]) with the following divide-and-conquer strategy has proven an effective choice.

1) *Problem decomposition:* splitting the object vectors into some smaller subcomponents.
2) *Optimize subcomponents:* evolve each subcomponent with a certain optimizer separately.
3) *Cooperative combination:* combine all subcomponents to form the whole system.

In [109], the authors proposed two DE-variants (DECC-I and DECC-II) that use self-adaptive NSDE (SaNSDE) (a synergy of the works reported in [108] and [76]) in a cooperative co-evolutionary framework with novel strategies for problem decomposition and subcomponents' cooperation. The algorithms were tested on a set of widely used benchmarks scaled up to 500 and 1000 dimensions. An important extension of the same work for better performance on rotated and non-separable high-dimensional functions has been reported in [110] where the authors use random grouping scheme with adaptive weighting for problem decomposition and coevolution. Some theoretical analysis is also presented in this paper to show why and how the new framework can be effective for optimizing large non-separable problems. The theoretical analysis illustrates how such strategies can help to

capture variable interdependencies in non-separable problems. Recently, Parsopoulos [S80] devised a cooperative micro-DE, which employs small cooperative subpopulations (with only few individuals) to detect subcomponents of the original problem's solution concurrently. The subcomponents are combined through cooperation of subpopulations to build complete solutions of the problem. Zamuda *et al*. [S81] proposed a DE-variant for large scale global optimization, where original DE is extended by log-normal self-adaptation of its control parameters and combined with cooperative co-evolution as a dimension decomposition mechanism.

Among the other approaches, Su presented a surrogate-assisted DE framework based on Gaussian process for solving large-scale computationally expensive problems in [S82]. Brest *et al.* [12] investigated a self-adaptive DE (abbreviated as jDEdynNP-F) where control parameters $F$ and $Cr$ are self-adapted and a population-size reduction method is used. The proposed jDEdynNP-F algorithm also employs a mechanism for sign changing of $F$ with some probability based on the fitness values of randomly chosen vectors, which are multiplied by $F$ in the mutation step of DE. The algorithm achieved third rank in CEC 2008 special session and competition on high-dimensional real-parameter optimization [98] that included non-separable functions like Schwefel's problem 2.21, Griewank's function, and fastfractal "doubledip" function.

### *D. DE for Optimization in Dynamic and Uncertain Environments*

In many real world applications, EAs often have to deal with optimization problems in the presence of a wide range of uncertainties. In general, there are four ways in which uncertainty may creep into the computing environment [S22]. First, the fitness function may be noisy. Second, the design variables and/or the environmental parameters may change after optimization, and the quality of the obtained optimal solution should be robust against environmental changes or deviations from the optimal point. Third, the fitness function may be approximated, which means that the fitness function suffers from approximation errors. Finally, the optimum of the problem to be solved changes its location over time and, thus, the optimizer should be able to track the optimum continuously. In all these cases, the EAs should be equipped with additional measures so that they are still able to work satisfactorily.

For a noisy problem, a deterministic choice of the scale factor and the greedy selection methods can be inadequate and a standard DE can easily fail at handling a noisy fitness function, as experimentally shown in [34]. Looking at the problem from a different perspective, the DE employs too much deterministic search logic for a noisy environment and therefore tends to stagnate. Das *et al.* [18] made an attempt to improve the performance of DE on noisy functions by first varying the scale factor randomly between 0.5 and 1 and secondly by incorporating two *not-so-greedy* selection mechanisms (threshold based selection and stochastic selection) in DE. Liu *et al*. [49] combined the advantages of the DE algorithm, the optimal computing budget allocation technique and simulated annealing (SA) algorithm to devise a robust hybrid DE method abbreviated as DEOSA) that can work well in noisy environments.

Mendes and Mohais presented DynDE [54]—a multi-population DE algorithm, developed specifically to optimize slowly time-varying objective functions. DynDE does not need any parameter control strategy for the $F$ or $Cr$. The main components in DynDE are as follows.

1) Usage of several populations in parallel.
2) Usage of uniform dither for $F$? [0, 1] as well as $Cr \in [0, 1]$.
3) To maintain diversity of the population based on two approaches.
    a) Reinitialization of a population if the best individual of a population gets too close to the best individual of another population. The population with the absolute best individual is kept while the other one is reinitialized. This way the various populations are prevented from merging.
    b) Randomization of one or more population vectors by adding a random deviation to the components.

Experimentally, the authors show that this new algorithm is capable of efficiently solving the moving peaks benchmark described by Branke [S83]. Very recently Brest *et al*. [13] investigated a self-adaptive DE algorithm (jDE) where $F$ and $Cr$ control parameters are self-adapted and a multi-population method with aging mechanism is used to handle dynamic fitness landscapes. This algorithm achieved the first rank in the Competition on "Evolutionary Computation in Dynamic and Uncertain Environments" in CEC2009 [45]. Angira and Santosh [5] presented a trigonometric differential evolution algorithm based on Fan and Lampinen's trigonometric mutation scheme [25] for solving dynamic optimization problems encountered in chemical engineering.

### *E. DE for Multimodal Optimization and Niching*

Many practical objective functions are highly multimodal and likely to have several high quality global and/or local solutions. Often, it is desirable to identify as many of these solutions as possible so that the most appropriate solution can be chosen. In order to identify many solutions of a multimodal optimization problem, several "niching" techniques have been developed. A niching method empowers an EA to maintain multiple groups within a single population in order to locate different optima. The niching techniques include crowding [103], fitness sharing [S203], [S209], clearing [S207], restricted tournament selection [81], [S204], and speciation [S205], [S208].

The crowding method [S206] allows competition for limited resources among similar individuals, i.e., within each niche. Generally, the similarity is measured using distance between individuals. The method compares an offspring with some randomly sampled individuals from the current population. The most similar individual will be replaced if the offspring is superior. Thomsen extended DE with a crowding scheme named as crowding DE (CDE) [103] to solve multimodal optimization problems. In CDE, when an offspring is generated, it will only compete with the most similar (measured by

Euclidean distance) individual in the population. The offspring will replace this individual if it has a better fitness value.

The fitness sharing method [S203], [S209] divides the population into different subgroups according to parameter space similarity of the individuals. An individual must share its information with other individuals within the same niche. The shared fitness for $i$th individual can be represented by using the following equation:

$$f_{shared}(i) = \frac{f_{original}(i)}{\sum_{j=1}^{N} sh(d_{ij})} \tag{37}$$

where the sharing function is calculated as

$$sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right)^{\alpha} & \text{if } d_{ij} < \sigma_{share} \\ 0 & \text{otherwise} \end{cases}$$

and $d_{ij}$ is the distance between individuals $i$ and $j$, $\sigma_{share}$ is the sharing radius, $N$ is the population size and $\alpha$ is a constant called sharing level. Thomsen integrated the fitness sharing concept with DE to form the sharing DE [103].

The restricted tournament selection (RTS) method uses tournament selection for multimodal optimization [S204]. The algorithm chooses a random sample of $w$ (window size) individuals from the population and determines the nearest one to the offspring, by using either Euclidean (for real variables) or Hamming (for binary variables) distance measure. The nearest member within the $w$ individuals will compete with the offspring and the one with higher fitness will survive in the next generation. The RTS was implemented with an ensemble of two different window sizes in [81] using the DE as the search method.

## VI. Analytical Studies on DE

Theoretical and empirical analyses of the properties of evolutionary algorithms are very important to understand their search behaviors and to develop more efficient algorithms [S84]. Compared to the plethora of works concerning the empirical study of parameter selection and tuning process in DE, not much research has so far been devoted to theoretically analyze the search mechanism and convergence properties of DE and this area remains largely open to prospective future research. Below, we discuss some of the analytical results so far obtained on DE.

### A. Population Variance and Explorative Power of DE

Some significant theoretical results on DE were first reported in [111] and then extended in [112] and [113] by Zaharie, where she theoretically analyzed the influence of the variation operators (mutation and recombination) and their parameters on the expected population variance. In [111], Zaharie showed that the expected population variance (after applying mutation and recombination, but without selection) of DE is greater than that of the ES algorithm analyzed in [S85]. This finding could explain to some extent the excellent performance of DE on certain test functions. In [114], Zaharie analyzed the impact on the expected population mean and variance of several variants of mutation and crossover operators used in DE algorithms. As a consequence of this analysis, she proposed a simple variance-based mutation operator without using differences but has the same impact on the population variance as classical DE operators proposed. She also presented a preliminary analysis of the distribution probability of the population in the case of a DE algorithm with binary encoding.

### B. Role of Crossover in DE

Very recently in [115], the influence of the crossover rate on the distribution of the number of mutated components and on the probability for a component to be taken from the mutant vector (mutation probability) is theoretically analyzed for several variants of crossover, including classical binomial and exponential strategies in DE. For each crossover variant the relationship between the crossover rate and the mutation probability is identified and its impact on the choice and adaptation of control parameters is analyzed both theoretically and numerically. With numerical experiments, the author illustrates the fact that the difference between binomial and exponential crossover variants is mainly due to different distributions of the number of mutated components. On the other hand, the behavior of exponential crossover variants was found to be more sensitive to the problem size than the behavior of variants based on binomial crossover.

### C. Evolutionary Search-Dynamics in DE

The first theoretical studies on the evolutionary search-dynamics of DE were carried out by Dasgupta *et al*. in [22] and [23]. The authors proposed a simple mathematical model of the underlying evolutionary dynamics of a 1-D DE-population (evolving with the DE/rand/1/bin algorithm) [22]. The model was based on the fact that DE perturbs each dimension separately and if a $D$-dimensional objective function is separable, this function can be optimized in a sequence of $D$ 1-D optimization processes. The model reveals that the fundamental dynamics of each search-agent (1-D parameter vector) in DE employs the gradient-descent type search strategy (although it uses no analytical expression for the gradient itself), with a learning rate parameter that depends on control parameters like scale factor $F$ and crossover rate $Cr$ of DE. It is due to the gradient descent type search strategy, that DE converges much faster than some variants of GA or PSO over uni-modal benchmarks [104]. The stability and convergence-behavior of the proposed dynamics was analyzed in the light of Lyapunov's stability theorems very near to the isolated equilibrium points during the final stages of the search in [23] and the rate of convergence on smooth uni-modal functions were found to largely depend on $Cr$. However, the analysis undertaken in [22] and [23] appeared to be of very limited scope from a practical point of view, as the authors considered a 1-D fitness landscape. Generalizing it for multi-dimensional search space can be a challenging future research work. Also proving the probabilistic convergence of DE on even very simple objective functions is still an open problem for the theorists working with EAs.

### D. Timing Complexity of DE

Runtime-complexity analysis of the population-based stochastic search techniques like DE is a critical issue by its own right. In [120], Zielinski *et al.* first investigated the runtime complexity of DE for various stopping criteria, both theoretically and empirically. The authors [120] pointed out that in each generation of DE a loop over $NP$ is conducted, containing a loop over $D$. Since the mutation and crossover operations are performed at the component level for each DE vector, the number of fundamental operations in DE/rand/1/bin is proportional to the total number of loops conducted until the termination of the algorithm. Thus, if the algorithm is stopped after a fixed number of generations $G_{max}$, then the runtime complexity is $O(NP \cdot D \cdot G_{\max})$. Moreover the authors also inferred that maximum distance criterion *MaxDist* yields best overall performance for the DE algorithm. Note that this criteria stops the execution of the algorithms if the maximum distance from every vector to the best population member is below a given threshold $m$ (say).

### E. Convergence of Multiobjective DE

Recently, Xue *et al*. [106] performed the mathematical modeling and convergence analysis of continuous multiobjective differential evolution under certain simplifying assumptions. The authors investigated the population evolution of the MODE with only reproduction operators, i.e., the differential mutation and crossover and assuming that that the DE-population is initialized by sampling from a Gaussian distribution with given mean and standard deviation. Using simple mathematics, they prove that the initial population $P_0$ is Gaussian distributed and contains the Pareto optimal set $\Lambda^*$, the subsequent populations generated by the MODE without selection are also Gaussian distributed and the population mean converges to the center of the Pareto optimal set $\Lambda^*$, i.e., if $\vec{X}_G$ be a solution vector belonging to the population $P_G$ at generation $G$, then

$$\lim_{G\to\infty} E(\vec{X}_G) = \vec{X}^* \tag{38}$$

where $\vec{X}^*$ is a random solution uniformly distributed on the probability support defined by $\Lambda^*$. The works were extended in [107] by modeling a discrete version of MODE, D-MODE, in the framework of Markov processes and the corresponding convergence properties were developed. However, in most practical situations with finite population size, the optimal solutions will not be present in the initial population. The exploration of MODE would identify the global optimal solution during the evolutionary process and the selection operator would keep those optimal solutions found in the evolution. Mathematical analysis of convergence under such situations is yet to be developed.

## VII. Engineering Applications of DE

Due to the rapidly growing popularity of DE as a simple and robust optimizer, researchers from several domains of science and engineering have been applying DE to solve optimization problems arising in their own fields. The literature on engineering applications of DE is huge and multifaceted. An internet search reveals that the number of DE research articles indexed in SCI database over the span of 2007–July 2009 is 3964 and out of these, there are more than thousands of application papers in diverse areas. For the sake of space economy, in Tables I and II we summarize only the major applications, where DE has been employed to solve the optimization problem, along with the type of the DE used, and the major publications associated with the application.

A keen observation of Tables I and II reveals that practitioners mostly prefer to use the classical DE schemes like DE/rand/1/bin, DE/target-to-best/1/bin, and so on for solving domain-specific problems. More research is necessary in order to investigate the applicability of the most state-of-the-art DE-variants (like SaDE [76], DEGL [21], and ODE [82]) outlined in Section IV for obtaining improved performances on practical problems. Specific applications may bear some properties that make it worthwhile revisiting or extending DE so that the optimization matches the problem in the best possible way. Generally any knowledge about the problem should be incorporated into the optimization method and/or the objective function in order to make it more efficient. The interested readers are redirected to appropriate references for details of the applications wherever necessary. Elaborations on some of the applications cited above are available in [71] and [78].

## VIII. Drawbacks of DE

Like all other metaheuristics, DE also has some drawbacks which we must take a look at before proceeding to the discussion on future research directions with DE.

Some of the recent publications [85], [S179] indicate that DE faces significant difficulty on functions that are not linearly separable and can be outperformed by CMA-ES. As pointed out by Sutton *et al*. [96], on such functions, DE must rely primarily on its differential mutation procedure, which, unlike its recombination strategy (with $Cr < 1$), is rotationally invariant. In [96], the authors also conjecture that this mutation strategy lacks sufficient selection pressure when appointing target and donor vectors to have satisfactory exploitative power on non-separable functions. The authors also propose a rank-based parent selection scheme to impose bias on the selection step, so that DE may also learn distribution information from elite individuals in the population and can thus sample the local topology of the fitness landscape better. However, they ended up with the opinion that much more research is necessary in this area to make DE sufficiently robust against the strong interdependency of the search variables. Experimenting with different selection procedures that may increase the generational selective pressure between parents and offspring may also serve as another avenue of future work.

In [44], Langdon and Poli made an attempt to evolve certain fitness landscapes with GP in order to demonstrate the benefits and weaknesses of a few population-based metaheuristics like PSO, DE, and CMA-ES. They pointed out that some problem landscapes may deceive DE such that it will get stuck in local optima most of the time; however, over similar landscapes

PSO will always find the global optima correctly within a maximum time-bound. The authors also indicated that DE sometimes has a limited ability to move its population large distances across the search space if the population is clustered in a limited portion of it. Indeed, in [S180, p. 20], the authors noted that the performance of DE deteriorates on the spiral "long path problem." The work in [44] also identified some landscape in which DE is outperformed by CMA-ES and a non-random gradient search based on Newton-Raphson's method (abbreviated as *N-R* in [S180]). A good amount of future research is needed to remove these drawbacks of DE. Also the most effective DE-variants developed so far should be investigated with the problem evolution methodology of Langdon and Poli, to identify their specific weak points over different function surfaces.

## IX. Potential Future Research Directions with DE

Although during the last ten years, research on and with DE has reached an impressive state, there are still many 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) Like many other EAs, the mutation schemes employed by DE are also additive and the donor (mutant), the scaled difference, and the target vectors lie in the same hyper-plane (see Fig. 2). Is it possible to think of a new rotation-based mutation operation where the base vector is first rotated in $D$-dimensional hyperspace and then perturbed with the scaled difference vector? In this case the rotation will be accomplished by pre-multiplying the base vector with a $D\times D$ linear transformation matrix Q and the donor vector will be formed as

$$\vec{V}_{i,G} = Q.\vec{X}_{r_1^i,G} + F \cdot (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}).$$

Consider a parameter vector $\vec{X}_{i,G}$ with each $x_{j,i,G} \in [-a, a]$ (say) for $j = 1, 2 \ldots D$. Then as per Section IV-D, each component of the opposite vector $\vec{X}^O_{i,G}$ is formed as $x^O_{j,i,G} = (-a + a) - x_{j,i,G} = -x_{j,i,G}$ and thus $\vec{X}^O_{i,G} = -\vec{X}_{i,G}$. Evidently, for symmetric search intervals, generation of an opposite vector amounts to rotating the actual vector by 180°. This technique has been used in ODE to improve the performance of DE. Hence, we propose to generalize this concept, i.e., rotating the mutant vectors at different angles, along with suitable self-adaptation schemes for the rotation matrix $Q$ to improve the explorative power and thus efficiency of DE to a large extent.

2) The effectiveness of conventional DE in solving a numerical optimization problem depends on the selected mutation strategy and its associated parameter values. However, different optimization problems require different mutation strategies with different parameter values depending on the nature of problem (unimodal and multimodal) and available computation resources. In addition, to solve a specific problem, different mutation strategies with different parameter settings may be better during different stages of the evolution than a single mutation strategy with unique parameter settings as in the conventional DE.
In the area of machine learning the concept of combining classifiers in an *ensemble* [S197] is employed to improve the overall classification performance. These classifiers could be based on a variety of classification methodologies. Similar concept was used [53] in conjunction with DE by forming an ensemble of mutation strategies and parameter values where a pool of mutation strategies, along with a pool of values corresponding to each associated parameter competes to produce successful offspring population. The candidate pool of mutation strategies and parameters should be restrictive to avoid the unfavorable influences of less effective mutation strategies and parameters. The mutation strategies or the parameters present in a pool should have diverse characteristics, so that they can exhibit distinct performance characteristics during different stages of the evolution, when dealing with a particular problem. This approach differs from SaDE as the latter adapts the parameter values slowly while the ensemble approach allows the parameters to jump to any appropriate value. This ensemble approach can be investigated further with enhanced mutation strategies and with different crossover approaches to solve different problem scenarios.
3) Future research may focus on integrating the opposition-number based initialization and generation jumping with self-adaptive DE variants like SaDE for improving the performance of the later. DEGL [21] puts forward the concept of topological neighborhood (ring shaped) of the population members for devising a local mutation scheme. The effect of various neighborhood topologies (star-shaped, wheel-shaped, fully connected, and so on) on the performance of DEGL should be investigated in future. Integrating DEGL in ensemble of DE schemes [S210], [52], [80], [100] should also be studied in the future.
4) Unlike the significant advancement made in the theoretical understanding of GAs, ES, and EP (see [S84], [S174]–[S176]) analysis of DE family of algorithms has still not made a considerable progress. Especially an investigation of the probabilistic convergence properties of DE even on some very simple objective functions remains largely an open problem so far. Concepts like drift analysis, martingale theory [S177] and stochastic Lyapunov energy functions [S178] that have already been applied to the convergence analysis of other real coded EAs may also be investigated in context to DE.
5) Sutton *et al*. conjectured in [96] that over DE's weak selective pressure (due to unbiased selection of parents or target vectors) may result in inefficient exploitation when relying on differential mutation, especially on the rotated optimization problems. The authors proposed a rank-based parent selection scheme. It is obtained by sorting the population by fitness and then drawing the indices for base vector and other vectors that constitute

TABLE I
SUMMARY OF APPLICATIONS OF DE TO ENGINEERING OPTIMIZATION PROBLEMS

| Sub Areas and Details | Types of DE Applied and References |
|---|---|
| Electrical Power Systems | |
| Economic dispatch | Chaotic DE [S31], hybrid DE with acceleration and migration [S87], DE/rand/1/bin [S88], hybrid DE with improved constraint handling [S89], variable scaling hybrid DE [S90] |
| Optimal power flow | DE/target-to-best/1/bin [S91], cooperative co-evolutionary DE [S92], DE/rand/1/bin with non-dominated sorting [S93], conventional DE/rand/1/bin [S94], [S96], DE with random localization [S95]. |
| Power system planning, generation expansion planning | Modified DE with fitness sharing [S97], conventional DE/rand/1/bin [S98], comparison of 10 DE strategies of Storn and Price [S99], robust searching hybird DE [S100] |
| Capacitor placement | Hybrid of ant system and DE [S49] |
| Distribution systems' network reconfiguration | Hybrid DE with variable scale factor [S101], mixed integer hybrid DE [S185]. |
| Power filter, power system stabilizer | Hybrid DE with acceleration and migration operators [S102], DE/target-to-best/1/bin [S103], hybrid of DE with ant systems [S104] |
| Electromagnetism, Propagation, and Microwave Engineering | |
| Capacitive voltage divider | MODE and NSDE [S105] |
| Electromagnetic inverse scattering | DE/rand/1/bin [S106], conventional DE with individuals in groups [S107], dynamic DE [77] |
| Design of circular waveguide mode converters | DE/rand/1/bin [S108] |
| Parameter estimation and property analysis for electromagnetic devices, materials, and machines | DE/rand/1/bin [S109]–[S111], [S113], DE/target-to-best/1/bin [S112] |
| Electromagnetic imaging | Conventional DE/rand/1/bin [S114], [S115], DE/best/1/bin [S116] |
| Antenna array design | Multimember DE (see [93] for details) [S117], hybrid real/integer-coded DE [S118], DE/rand/1/bin [S119], [S120], modified DE with refreshing distribution operator and fittest individual refinement operator [S121], DE/best/1/bin [S122], MOEA/D-DE [68], [69] |
| Control Systems and Robotics | |
| System identification | Conventional DE/rand/1/bin [S123]–[S126] |
| Optimal control problems | DE/rand/1/bin and DE/best/2/bin [S127], modified DE with adjustable control weight gradient methods [S128]. |
| Controller design and tuning | Self adaptive DE [S129], DE/rand/1 with arithmetic crossover [S130], DE/rand/1/bin with random scale factor and time-varying *Cr* [S131]. |
| Aircraft control | Hybrid DE with downhill simplex local optimization [55]. |
| Nonlinear system control | Hybrid DE with convex mutation [15]. |
| Simultaneous localization and modeling problem | DE/rand/1/bin [S132], [S133] |
| Robot motion planning and navigation | DE/rand/1/bin [S134], [S135] |
| Cartesian robot control | Memetic compact DE [61] |
| Multi-sensor data fusion | DE/best/2/bin [S136] |

TABLE II
SUMMARY OF APPLICATIONS OF DE TO ENGINEERING OPTIMIZATION PROBLEMS (CONTINUED FROM TABLE I)

| Sub Areas and Details | Types of DE Applied and References |
|---|---|
| Bioinformatics | |
| Gene regulatory networks | DE with adaptive local search (see [22] for details) [63], hybrid of DE and PSO [S137] |
| Micro-array data analysis | Multiobjective DE-variants (MODE, DEMO) [S138] |
| Protein folding | DE/rand/1/bin [S139] |
| Bioprocess optimization | DE/rand/1/bin [S140] |
| Chemical Engineering | |
| Chemical process synthesis and design | Modified DE with single array updating [S141], [7], 10 DE-variants of Storn and Price (see [74], [75]) compared in [S142], [S144], multiobjective DE [S143], hybrid DE with migration and acceleration operators [S145]. |
| Phase equilibrium and phase study | DE/rand/1/bin [S146]. |
| Parameter estimation of chemical process | Hybrid DE with geometric mean mutation [S147], DE/target-to-best/1/exp [S148]. |
| Pattern Recognition and Image Processing | |
| Data clustering | DE/rand/1/bin [S149], DE with random scale factor and time-varying crossover rate [20], DE with neighborhood-based mutation [S150] |
| Pixel clustering and region-based image segmentation | Modified DE with local and global best mutation [S151], DE with random scale factor and time-varying crossover rate [S152]. |
| Feature extraction | DE/rand/1/bin [S153] |
| Image registration and enhancement | DE/rand/1/bin [S154], DE with chaotic local search [S155] |
| Image Watermarking | DE/rand/1/bin and DE/target-to-best/1/bin [S156] |
| Artificial Neural Networks | |
| Training of feed-forward ANNs | DE/rand/1/bin [S157], [S160], generalization-based DE (GDE) [S158], DE/target-to-best/1/bin [S159] |
| Training of wavelet neural networks | DE/rand/1/bin [S161] |
| Training of B-Spline neural networks | DE with chaotic sequence-based adjustment of scale factor $F$ [S162] |
| Signal Processing | |
| Non-linear $\tau$ estimation | Dynamic DE [S163] |
| Digital filter design | DE/rand/1/bin [S164], [S165], DE with random scale factor [S166] |
| Fractional order signal processing | DE/rand/1/bin [S167] |
| Others | |
| Layout synthesis for MEMS | Improved DE with stochastic ranking (SR) [S168] |
| Engineering design | DE/rand/1/bin [S169], multimember constraint adaptive DE [93] |
| Manufacturing optimization | DE/rand/1/bin [S170], hybrid DE with forward/backward transformation [S171] |
| Molecular configuration | Local search-based DE [S172] |
| Urban energy management | Hybrid of DE and CMA-ES [S173] |
| Optoelectronics | DE/target-to-best/1/bin [S186] |
| Chess evaluation function tuning | DE/rand/1/bin [S228] |

the difference vectors from a linear distribution function [S198], such that the top ranked individual is more likely to be selected than the one with poorer rank. Since high selective pressure can often cause rapid loss of diversity of the population, in order to obtain a better balance between exploration and exploitation (even when the parent selection is biased), a time-varying selection pressure is necessary. One interesting way to achieve this is to introduce an annealing schedule to the bias controlling term of the linear distribution function, such that at the beginning stage of the search the probability of selecting a top ranked individual is small (thus favoring exploration) and it gradually grows with the intensification of the search so that good exploitation can be achieved during the later stages. Also it is worthy to investigate the effect of tournament selection instead of rank-based selection that necessitates the sorting of the population at each generation.

6) In an EA, exploration is directly connected with enabling the algorithm to advance from one optimum to another. In the EA-literature, a less common exploration mechanism is *saddle crossing* [S199], which consists in gradual change of the position of the whole population along the saddle that connects two adjacent attraction basins of local maxima. Saddle crossing usually takes a few generations and is accompanied with a temporary decrease of the mean population fitness. Fitness proportionate selection without elitism is reported to be the most appropriate to promote saddle crossing, since is implies relatively soft selection [S199]. DE usually employs a very hard selection criterion with one-to-one competition between a parent and its offspring. For this reason, DE is very sensitive to the choice of the initial population and may suffer from the premature convergence [94]. A very interesting future research topic may be to integrate the mutation operator taken from DE with the non-elitist, fitness proportionate selection, without crossover that can result in good saddle crossing abilities of an EA [S200]. This can significantly reduce the risk of premature convergence, even in the case of unfortunate population initialization nearby a local optimum with a large attraction basin.
7) Combinatorial optimization problems deal with discrete parameters and can be found profusely in diverse domains of engineering. As pointed out in Section IV-H, DE already achieved some reputation of solving discrete, binary and mixed-integer problems. However, there is no good evidence so far that DE is applicable to strict-sense combinatorial problems, especially when they are heavily constrained. In [74], the authors discussed the topic of combinatorial problems and they attributed the success of DE-based solutions to combinatorial problems to well-chosen repair mechanisms in the algorithm rather than DE-mutation. Therefore, proving the applicability of DE to strict-sense combinatorial problems remains an open problem so far. Finding a discrete operator that corresponds to the difference vector in the continuous domain is also a challenging issue for future research. Moreover, in discrete DE-variants, it is required that the combination of a base vector and a difference vector (or recombination vector) yields a new valid vector. The validity of the newly generated vector is a big problem for most of the classical combinatorial problems like the traveling salesperson problem.
8) Many objective optimization problems [S181], [S182] typically deal with more than three objective functions. Many conventional MOEAs applying Pareto optimality as a ranking metric may perform poorly over a large number of objective functions. Extending the multiobjective variants of DE to solve many-objective problems remains open as a challenging field of future research so far.

## X. Conclusion

With the increasing complexity of real world optimization problems, demand for robust, fast, and accurate optimizers is on the rise among researchers from various fields. DE emerged as a simple and efficient scheme for global optimization over continuous spaces more than a decade ago. Over the past few years, many researchers have contributed to make it a general and fast optimization method for any kind of objective function by twisting and tuning the various constituents of DE, i.e., initialization, mutation, diversity enhancement, and selection of DE as well as by the choice of the control variables, even though the NFL [S25] suggested already that such a cure-all-optimization algorithm could not exist. Nonetheless the existing literature clearly indicates that DE exhibits remarkable performance in optimizing a wide variety of multi-dimensional, multiobjective and multimodal optimization problems.

This paper attempted to provide an overall picture of the state-of-the-art research on and with DE. Starting with a comprehensive introduction to the basic steps of the DE algorithm, it discussed the different schemes of parameter control and adaptation for DE and then overviewed several promising variants of the conventional DE. Next it provided an extensive review of the modifications of DE for tackling constrained, multiobjective, uncertain, and large-scale optimization problems. It gave a brief overview of various most significant engineering applications of DE and unfolded a number of future research directions as well. The content of the paper indicates the fact that DE will continue to remain a vibrant and active field of multi-disciplinary research in the years to come.

## References

[1] H. Abbass, "The self-adaptive Pareto differential evolution algorithm," in *Proc. Congr. Evol. Comput.*, vol. 1. May 2002, pp. 831–836.
[2] H. A. Abbass and R. Sarker, "The Pareto differential evolution algorithm," *Int. J. Artif. Intell. Tools*, vol. 11, no. 4, pp. 531–552, 2002.
[3] M. M. Ali and A. Törn, "Population set based global optimization algorithms: Some modifications and numerical studies," *Comput. Oper. Res.*, vol. 31, no. 10, pp. 1703–1725, 2004.
[4] M. M. Ali and Z. Kajee-Bagdadi, "A local exploration-based differential evolution algorithm for constrained global optimization," *Appl. Math. Comput.*, vol. 208, no. 1, pp. 31–48, Feb. 2009.
[5] R. Angira and A. Santosh, "Optimization of dynamic systems: A trigonometric differential evolution approach," *Comput. Chem. Eng.*, vol. 31, no. 9, pp. 1055–1063, Sep. 2007.

[6] B. V. Babu and M. M. L. Jehan, "Differential evolution for multiobjective optimization," in *Proc. Congr. Evol. Comput.*, vol. 4. Dec. 2003, pp. 2696–2703.

[7] B. V. Babu and R. Angira, "Modified differential evolution (MDE) for optimization of non-linear chemical processes," *Comput. Chem. Eng.*, vol. 30, nos. 6–7, pp. 989–1002, 2006.

[8] H.-G. Beyer and K. Deb, "On self-adapting features in real-parameter evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 5, no. 3, pp. 250–270, Jun. 2001.

[9] A. Biswas, S. Dasgupta, S. Das, and A. Abraham, "A synergy of differential evolution and bacterial foraging algorithm for global optimization," *Neural Netw. World*, vol. 17, no. 6, pp. 607–626, 2007.

[10] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[11] J. Brest and M. S. Mauèec, "Population size reduction for the differential evolution algorithm," *Appl. Intell.*, vol. 29, no. 3, pp. 228–247, Dec. 2008.

[12] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, "High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2008, pp. 2032–2039.

[13] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, "Dynamic optimization using self-adaptive differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, May 2009, pp. 415–422.

[14] A. Caponio, F. Neri, and V. Tirronen, "Super-fit control adaptation in memetic differential evolution frameworks," *Soft Comput. A Fusion Found. Methodol. Applicat.*, vol. 13, no. 8, pp. 811–831, 2009.

[15] C.-H. Chen, C.-J. Lin, and C.-T. Lin, "Nonlinear system control using adaptive neural fuzzy networks based on a modified differential evolution," *IEEE Trans. Syst. Man Cybern. Part C*, vol. 39, no. 4, pp. 459–473, Jul. 2009.

[16] S. Das, A. Konar, and U. K. Chakraborty, "Improving particle swarm optimization with differentially perturbed velocity," in *Proc. Genet. Evol. Comput. Conf.*, Jun. 2005, pp. 177–184.

[17] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," in *Proc. ACM-SIGEVO GECCO*, Jun. 2005, pp. 991–998.

[18] S. Das, A. Konar, and U. Chakraborty, "Improved differential evolution algorithms for handling noisy optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. 2005, pp. 1691–1698.

[19] S. Das, A. Konar, and U. K. Chakraborty, "Annealed differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, 2007, pp. 1926–1933.

[20] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Trans. Syst. Man Cybern. Part A*, vol. 38, no. 1, pp. 218–236, Jan. 2008.

[21] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553, Jun. 2009.

[22] S. Dasgupta, S. Das, A. Biswas, and A. Abraham, "The population dynamics of differential evolution: A mathematical model," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2008, pp. 1439–1446.

[23] S. Dasgupta, S. Das, A. Biswas, and A. Abraham, "On stability and convergence of the population-dynamics in differential evolution," *AI Commun.*, vol. 22, no. 1, pp. 1–20, 2009.

[24] K. Deb and H.-G. Beyer, "Self-adaptive genetic algorithms with simulated binary crossover," *Evol. Comput.*, vol. 9, no. 2, pp. 197–221, Jun. 2001.

[25] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *J. Global Optimization*, vol. 27, no. 1, pp. 105–129, 2003.

[26] V. Feoktistov and S. Janaqi, "Generalization of the strategies in differential evolution," in *Proc. 18th IPDPS*, Apr. 2004, p. 165a.

[27] Y. Gao and Y. Wang, "A memetic differential evolutionary algorithm for high dimensional function spaces optimization," in *Proc. 3rd ICNC 20*, vol. 4. Aug. 2007, pp. 188–192.

[28] V. L. Huang, A. K. Qin, and P. N. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2006, pp. 324–331.

[29] V. L. Huang, A. K. Qin, P. N. Suganthan, and M. F. Tasgetiren, "Multiobjective optimization based on self-adaptive differential evolution algorithm," in *Proc. Congr. Evol. Comput.*, Sep. 2007, pp. 3601–3608.

[30] V. L. Huang, S. Z. Zhao, R. Mallipeddi, and P. N. Suganthan, "Multiobjective optimization using self-adaptive differential evolution algorithm (special session and competition on 'performance assessment of constrained/bound constrained multiobjective optimization algorithms')," in *Proc. Conf. Congr. Evol. Comput.*, May 2009, pp. 190–194.

[31] F. Huang, L. Wang, and Q. He, "An effective co-evolutionary differential evolution for constrained optimization," *Appl. Math. Comput.*, vol. 186, no. 1, pp. 340–356, Mar. 2007.

[32] A. W. Iorio and X. Li, "Solving rotated multiobjective optimization problems using differential evolution," in *Proc. AI: Adv. Artif. Intell.*, LNCS 3339. 2004, pp. 861–872.

[33] P. Kaelo and M. M. Ali, "Differential evolution algorithms using hybrid mutation," *Comput. Optimization Applicat.*, vol. 37, pp. 231–246, Jun. 2007.

[34] T. Krink, B. Filipič, and G. B. Fogel, "Noisy optimization problems: A particular challenge for differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, 2004, pp. 332–339.

[35] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of generalized differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1. Sep. 2005, pp. 443–450.

[36] S. Kukkonen and J. Lampinen, "Constrained real-parameter optimization with generalized differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2006, pp. 911–918.

[37] J. Lampinen, "A bibliography of differential evolution algorithm," Lab. Inform. Process., Dept. Inform. Technol., Lappeenranta Univ. Technol., Lappeenranta, Finland, Tech. Rep., 1999 [Online]. Available: http://www.lut.fi/∼jlampine/debiblio.htm

[38] J. Lampinen, "Differential evolution: New naturally parallel approach for engineering design optimization," in *Developments in Computational Mechanics with High Performance Computing*, B. H. V. Topping, Ed. Edinburgh, U.K.: Civil-Comp Press, 1999, pp. 217–228.

[39] J. Lampinen, "A constraint handling approach for the differential evolution algorithm," in *Proc. Congr. Evol. Comput.*, vol. 2. May 2002, pp. 1468–1473.

[40] J. Lampinen and I. Zelinka, "Mixed integer-discrete-continuous optimization with differential evolution," in *Proc. 5th Int. Mendel Conf. Soft Comput.*, Jun. 1999, pp. 71–76.

[41] J. Lampinen and I. Zelinka, "On stagnation of the differential evolution algorithm," in *Proc. 6th Int. Mendel Conf. Soft Comput.*, Jun. 2000, pp. 76–83.

[42] R. Landa Becerra and C. A. Coello Coello, "Solving hard multiobjective optimization problems using $\varepsilon$-constraint with cultured differential evolution," in *Proc. 9th Int. Conf. Parallel Problem Solving Nature*, LNCS 4193. Sep. 2006, pp. 543–552.

[43] R. Landa Becerra and C. A. Coello Coello, "Cultured differential evolution for constrained optimization," *Comput. Methods Appl. Mech. Eng.*, vol. 195, nos. 33–36, pp. 4303–4322, Jul. 2006.

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

[45] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan, "Benchmark generator for CEC'2009 competition on dynamic optimization," Univ. Leicester, Leicester, U.K., Univ. Birmingham, U.K., Nanyang Technological Univ., Singapore, Tech. Rep., Sep. 2008.

[46] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, Apr. 2009.

[47] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. Coello Coello, and K. Deb, "Problem definitions and evaluation criteria for the CEC 2006 (special session on constrained real-parameter optimization)," Nanyang Technol. Univ., Singapore, Tech. Rep., 2006.

[48] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Comput. A Fusion Founda. Methodol. Applicat.*, vol. 9, no. 6, pp. 448–462, 2005.

[49] B. Liu, X. Zhang, and H. Ma, "Hybrid differential evolution for noisy optimization," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2008, pp. 587–592.

[50] R. Mallipeddi and P. N. Suganthan, "Empirical study on the effect of population size on differential evolution algorithm," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2008, pp. 3663–3670.

[51] R. Mallipeddi and P. N. Suganthan, "Differential evolution algorithm with ensemble of populations for global numerical optimization," *OPSEARCH*, vol. 46, no. 2, pp. 184–213, Jun. 2009.

[52] R. Mallipeddi and P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 561–579, Aug. 2010.

[53] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of param-

eters and mutation strategies," *Appl. Soft Comput.*, to be published.

[54] R. Mendes and A. S. Mohais, "DynDE: A differential evolution for dynamic optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. 2005, pp. 2808–2815.

[55] P. P. Menon, J. Kim, D. G. Bates, and I. Postlethwaite, "Clearance of nonlinear flight control laws using hybrid evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 689–699, Dec. 2006.

[56] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *Proc. Genet. Evol. Comput. Conf.*, 2006, pp. 485–492.

[57] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "Modified differential evolution for constrained optimization," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2006, pp. 332–339.

[58] P. W. Moore and G. K. Venayagamoorthy, "Evolving digital circuit using hybrid particle swarm optimization and differential evolution," *Int. J. Neural Syst.*, vol. 16, no. 3, pp. 163–177, 2006.

[59] F. Neri and V. Tirronen, "Scale factor local search in differential evolution," *Memetic Comput.*, vol. 1, no. 2, pp. 153–171, Jun. 2009.

[60] F. Neri and V. Tirronen, "Recent advances in differential evolution: A review and experimental analysis," *Artif. Intell. Rev.*, vol. 33, no. 1, pp. 61–106, Feb. 2010.

[61] F. Neri and E. Mininno, "Memetic compact differential evolution for Cartesian robot control," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 54–65, May 2010.

[62] N. Noman and H. Iba, "Enhancing differential evolution performance with local search for high dimensional function optimization," in *Proc. Conf. Genet. Evol. Comput.*, Jun. 2005, pp. 967–974.

[63] N. Noman and H. Iba, "Inferring gene regulatory networks using differential evolution with local search heuristics," *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 4, no. 4, pp. 634–647, Oct. 2007.

[64] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 107–125, Feb. 2008.

[65] M. G. H. Omran, A. Salman, and A. P. Engelbrecht, "Self-adaptive differential evolution," in *Proc. Comput. Intell. Security*, Lecture Notes in Artificial Intelligence 3801. 2005, pp. 192–199.

[66] M. G. H. Omran, A. P. Engelbrecht, and A. Salman, "Bare bones differential evolution," *Eur. J. Oper. Res.*, vol. 196, no. 1, pp. 128–139, Jul. 2009.

[67] G. Pampara, A. P. Engelbrecht, and N. Franken, "Binary differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2006, pp. 1873–1879.

[68] S. Pal, Q. Boyang, S. Das, and P. N. Suganthan, "Optimal synthesis of linear antenna arrays with multiobjective differential evolution," *Prog. Electromag. Res. PIERB*, vol. 21, pp. 87–111, 2010.

[69] S. Pal, S. Das, A. Basak, and P. N. Suganthan, "Synthesis of difference patterns for monopulse antennas with optimal combination of array-size and number of subarrays: A multiobjective optimization approach," *Prog. Electromag. Res. PIERB*, vol. 21, pp. 257–280, 2010.

[70] K. E. Parsopoulos, D. K. Taoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Vector evaluated differential evolution for multiobjective optimization," in *Proc. Congr. Evol. Comput.*, vol. 1. Jun. 2004, pp. 204–211.

[71] V. P. Plagianakos, D. K. Tasoulis, and M. N. Vrahatis, "A review of major application areas of differential evolution," in *Advances in Differential Evolution*, SCI 143, U. K. Chakraborty, Ed. Berlin, Germany: Springer, 2008, pp. 197–238.

[72] K. V. Price, "Differential evolution vs. the functions of the 2nd ICEO," in *Proc. IEEE Int. Conf. Evol. Comput.*, Apr. 1997, pp. 153–157.

[73] K. V. Price and R. Storn, "Differential evolution: A simple evolution strategy for fast optimization," *Dr. Dobb's J.*, vol. 22, no. 4, pp. 18–24, 1997.

[74] K. Price, R. Storn, and J. Lampinen, *Differential Evolution—A Practical Approach to Global Optimization*. Berlin, Germany: Springer, 2005.

[75] K. V. Price, "An introduction to differential evolution," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and V. Glover, Eds. London, U.K.: McGraw-Hill, 1999, pp. 79–108.

[76] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

[77] A. Qing, "Dynamic differential evolution strategy and applications in electromagnetic inverse scattering problems," *IEEE Trans. Geosci. Remote Sensing*, vol. 44, no. 1, pp. 116–125, Jan. 2006.

[78] A. Qing, *Differential Evolution—Fundamentals and Applications in Electrical Engineering*. New York: Wiley, Sep. 2009.

[79] B. Y. Qu and P. N. Suganthan, "Multiobjective evolutionary algorithms based on the summation of normalized objectives and diversified selection," *Inform. Sci.*, vol. 180, no. 17, pp. 3170–3181, Sep. 2010.

[80] B. Y. Qu and P. N. Suganthan, "Constrained multiobjective optimization algorithm with ensemble of constraint handling methods," *Eng. Optimization*, to be published.

[81] B. Y. Qu and P. N. Suganthan, "Novel multimodal problems and differential evolution with ensemble of restricted tournament selection," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2010, pp. 3480–3486.

[82] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64–79, Feb. 2008.

[83] T. Robic and B. Filipic, "DEMO: Differential evolution for multiobjective optimization," in *Proc. 3rd Int. Conf. Evol. Multi-Criterion Optimization*, LNCS 3410. 2005, pp. 520–533.

[84] J. Rönkkönen and J. Lampinen, "On using normally distributed mutation step length for the differential evolution algorithm," in *Proc. 9th Int. Conf. Soft Comput. MENDEL*, 2003, pp. 11–18.

[85] J. Ronkkonen, S. Kukkonen, and K. V. Price, "Real parameter optimization with differential evolution," in *Proc. IEEE CEC*, vol. 1. 2005, pp. 506–513.

[86] L. V. Santana-Quintero and C. A. Coello Coello, "An algorithm based on differential evolution for multiobjective problems," *Int. J. Comput. Intell. Res.*, vol. 1, no. 2, pp. 151–169, 2005.

[87] L. V. Santana-Quintero, A. G. Hernández-Díaz, J. Molina, C. A. Coello Coello, and R. Caballero, "DEMORS: A hybrid multiobjective optimization algorithm using differential evolution and rough sets for constrained problems," *Comput. Oper. Res.*, vol. 37, no. 3, pp. 470–480, Mar. 2010.

[88] 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

[89] R. Storn and K. V. Price, "Minimizing the real functions of the ICEC 1996 contest by differential evolution," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1996, pp. 842–844.

[90] R. Storn, "On the usage of differential evolution for function optimization," in *Proc. North Am. Fuzzy Inform. Process. Soc.*, 1996, pp. 519–523.

[91] R. Storn, "Differential evolution design of an IIR-filter with requirements for magnitude and group delay," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1996, pp. 268–273.

[92] 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.

[93] R. Storn, "System design by constraint adaptation and differential evolution," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 22–34, Apr. 1999.

[94] R. Storn, "Differential evolution research: Trends and open questions," in *Advances in Differential Evolution*, U. K. Chakraborty, Ed. Berlin, Germany: Springer, 2008, pp. 1–32.

[95] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Nanyang Technol. Univ., Singapore, Tech. Rep., IIT Kanpur, Kanpur, India, KanGAL Rep. #2005005, May 2005.

[96] A. M. Sutton, M. Lunacek, and L. D. Whitley, "Differential evolution and non-separability: Using selective pressure to focus search," in *Proc. 9th Annu. Conf. GECCO*, Jul. 2007, pp. 1428–1435.

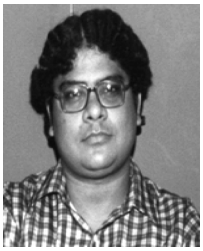
[97] T. Takahama and S. Sakai, "Constrained optimization by the $\varepsilon$ constrained differential evolution with gradient-based mutation and feasible elites," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2006, pp. 308–315.

[98] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang, "Benchmark functions for the CEC'2008 special session and competition on large scale global optimization," Nature Inspired Comput. Applicat. Lab., USTC, China, Nanyang Technol. Univ., Singapore, Tech. Rep., 2007.

[99] M. F. Tasgetiren and P. N. Suganthan, "A multi-populated differential evolution algorithm for solving constrained optimization problem," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2006, pp. 340–354.

[100] M. F. Tasgetiren, P. N. Suganthan, and Q. K. Pan, "An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem," *Appl. Math. Comput.*, vol. 215, no. 9, pp. 3356–3368, Jan. 2010.

[101] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Parallel differential evolution," in *Proc. Congr. Evol. Comput.*, 2004, pp. 2023–2029.
[102] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Comput. A Fusion Found. Methodol. Applicat.*, vol. 10, no. 8, pp. 673–686, 2006.
[103] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, 2004, pp. 1382–1389.
[104] J. Vesterstrøm and R. A. Thomson, "Comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proc. IEEE Congr. Evol. Comput.*, 2004, pp. 1980–1987.
[105] F. Xue, A. C. Sanderson, and R. J. Graves, "Pareto-based multiobjective differential evolution," in *Proc. Congr. Evol. Comput.*, vol. 2. 2003, pp. 862–869.
[106] F. Xue, A. C. Sanderson, and R. J. Graves, "Modeling and convergence analysis of a continuous multiobjective differential evolution algorithm," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1. Sep. 2005, pp. 228–235.
[107] F. Xue, A. C. Sanderson, and R. J. Graves, "Multiobjective differential evolution: Algorithm, convergence analysis, and applications," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1. Sep. 2005, pp. 743–750.
[108] Z. Yang, J. He, and X. Yao, "Making a difference to differential evolution," in *Advances in Metaheuristics for Hard Optimization*, Z. Michalewicz and P. Siarry, Eds. Berlin, Germany: Springer, 2007, pp. 415–432.
[109] Z. Yang, K. Tang, and X. Yao, "Differential evolution for high-dimensional function optimization," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 3523–3530.
[110] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inform. Sci.*, vol. 178, no. 15, pp. 2985–2999, 2008.
[111] D. Zaharie, "On the explorative power of differential evolution," in *Proc. 3rd Int. Workshop Symbolic Numerical Algorithms Scientific Comput.*, Oct. 2001 [Online]. Available: http://web.info.uvt.ro/∼dzaharie/online_papers.html
[112] D. Zaharie, "Critical values for the control parameters of differential evolution algorithms," in *Proc. 8th Int. Mendel Conf. Soft Comput.*, 2002, pp. 62–67.
[113] D. Zaharie, "Parameter adaptation in differential evolution by controlling the population diversity," in *Proc. 4th Int. Workshop Symbolic Numeric Algorithms Sci. Comput.*, 2002, pp. 385–397.
[114] D. Zaharie, "Statistical properties of differential evolution and related random search algorithms," in *Proc. Int. Conf. Comput. Statist.*, Aug. 2008, pp. 473–485.
[115] D. Zaharie, "Influence of crossover on the behavior of differential evolution algorithms," *Appl. Soft Comput.*, vol. 9, no. 3, pp. 1126–1138, Jun. 2009.
[116] A. Zamuda, J. Brest, B. Boskovic, and V. Zumer, "Differential evolution for multiobjective optimization with self adaptation," in *Proc. Congr. Evol. Comput.*, Sep. 2007, pp. 3617–3624.
[117] M. Zhang, W. Luo, and X. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," *Inform. Sci.*, vol. 178, no. 15, pp. 3043–3074, Aug. 2008.
[118] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.
[119] W.-J. Zhang and X.-F. Xie, "DEPSO: Hybrid particle swarm with differential evolution operator," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2003, pp. 3816–3821.
[120] K. Zielinski, D. Peters, and R. Laur, "Run time analysis regarding stopping criteria for differential evolution and particle swarm optimization," in *Proc. 1st Int. Conf. Exp./Process/System Modelling/Simulation/Optimization*, 2005 [Online]. Available: http://www.item.uni-bremen.de/research/papers/paper.pdf/Zielinski.Karin/zielinski-i05run.pdf
[121] K. Zielinski and R. Laur, "Constrained single-objective optimization using differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2006, pp. 927–934.



**Swagatam Das** (M'10) received the B.E. Tel. E., M.E. Tel. E. (control engineering specialization), and Ph.D. degrees, all from Jadavpur University, Kolkata, India, in 2003, 2005, and 2009, respectively.

He is currently an Assistant Professor with the Department of Electronics and Telecommunication Engineering, Jadavpur University. He has published more than 100 research articles in peer-reviewed journals and international conferences. He has co-authored a research monograph on metaheuristic clustering techniques published by Springer, Berlin, Germany, in 2009. His current research interests include evolutionary computing, swarm intelligence, pattern recognition, bioinformatics, control systems engineering, and digital signal processing.

Dr. Das serves as an Associate Editor for the *Information Sciences Journal* (Elsevier), and as an Editorial Board Member of the *International Journal of Artificial Intelligence and Soft Computing* and the *International Journal of Adaptive and Autonomous Communication Systems*. He is a Founding Co-Editor-in-Chief of *Swarm and Evolutionary Computation*, an international journal from Elsevier. He has been acting as a Regular Reviewer for journals like *Pattern Recognition*, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS, and IEEE TRANSACTIONS ON SMC Part A and Part B.

**Ponnuthurai Nagaratnam Suganthan** (M'91–SM'01) received the B.A. degree, the Postgraduate Certificate, and the M.A. degree in electrical and information engineering from the University of Cambridge, Cambridge, U.K., in 1990, 1992, and 1994, respectively, and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

He was a Pre-Doctoral Research Assistant with the Department of Electrical Engineering, University of Sydney, Sydney, Australia, from 1995 to 1996, and was a Lecturer with the Department of Computer Science and Electrical Engineering, University of Queensland, Brisbane, Australia, from 1996 to 1999. Since 1999, he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University, where he was previously an Assistant Professor and is currently an Associate Professor. His current research interests include evolutionary computation, pattern recognition, multi-objective evolutionary algorithms, bioinformatics, applications of evolutionary computation, and neural networks.

Dr. Suganthan is an Associate Editor of the journals IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, *Information Sciences*, *Pattern Recognition*, and the *International Journal of Swarm Intelligence Research*. He is a Founding Co-Editor-in-Chief of *Swarm and Evolutionary Computation*, an international journal from Elsevier.