Sequence-based Deterministic Initialization for Evolutionary Algorithms

Saber Elsayed[1], Ruhul Sarker[1] and Carlos Coello Coello[2]

***Abstract*—It is well known that the performances of evolutionary algorithms are influenced by the quality of their initial populations. Over the years, many different techniques for generating an initial population by uniformly covering as much of the search space as possible have been proposed. However, none of these approaches considers any input from the function that must be evolved using that population. In this paper, a new initialization technique, which can be considered a heuristic space-filling approach, based on both the function to be optimized and the search space, is proposed. It was tested on two well-known unconstrained sets of benchmark problems using several computational intelligence algorithms. The results obtained reflected its benefits as the performances of all these algorithms were significantly improved compared with those of the same algorithms with currently available initialization techniques. The new technique also proved its capability to provide useful information about the function's behaviour and, for some test problems, the initial population produced high-quality solutions. This method was also tested on a few multi-objective problems, with the results demonstrating its benefits.**

***Keywords*—evolutionary algorithms, population initialization, experimental design**

I. INTRODUCTION

SOLVING optimization problems involves finding the values of their decision variables so that one or more objective functions is optimized (either maximized or minimized) [1]. Generally, these problems may possess different mathematical properties, i.e., different types of variables, and their objective function can be linear or nonlinear, continuous or discontinuous and uni- or multi-modal. They can be found in many fields including, but not limited to, science, engineering and business [2]. Mathematically speaking, they can be formulated as

$$\text{minimize or maximize } f(\overrightarrow{x})$$

$$\text{subject to: } \underline{x}_j \leq x_j \leq \overline{x}_j,\ j=1,2,\ldots,D \tag{1}$$

where$f(\overrightarrow{x})$ is the objective function, $\overrightarrow{x}=[x_0, x_2, ..., x_D]$ a vector with D decision variables with each x_j has lower and upper limits $\underline{x}_j$ and $\overline{x}_j$, respectively.

Depending on a problem's complexity, researchers and practitioners choose either traditional optimization approaches [3] or computational intelligence (CI) methods to solve it. Of the CI methods, evolutionary algorithms (EAs), such as genetic algorithms (GAs) [4], differential evolution (DE) [5] and evolution strategy (ES) [6], as well as swarm intelligence approaches, such as particle swarm optimization (PSO) [7] are very popular. EAs, the type of approach considered in this paper, have many advantages over traditional optimization techniques [8], including the fact that they do not impose specific mathematical requirements (e.g., differentiability). Also, they operate on a set of solutions, instead of one at a time, which minimizes the chance of their becoming trapped in local optima, and work well in the presence of noise and in dynamic environments. However, they also have some drawbacks [8], including that: (1) in general, they cannot be guaranteed to produce the global optimum (i.e., they can become stuck in a local optimum); (2) their performances are sensitive to their parameters which are problem dependent; and (3) they do not normally exploit any domain information that may be available.

Over the years, researchers have attempted to overcome these drawbacks by proposing (1) approaches for maintaining diversity within the population [9]; (2) self-adaptive techniques for automatically determining their control parameters [10], [11]; and (3) frameworks that combine more than one EA or operator [12] [13]–[16].

Due to the influence of the initialization step in the performance of an EA, recently, a rapidly increasing number of new initialization techniques has been proposed, with their main motivation to uniformly cover the search space. They claim an increase in the probability of finding global optima and/or a reduction in computational costs for obtaining near-optimal solutions, i.e., an increase in the convergence rate, or both [17], [18] [19]. Based on a recent research study [19], existing initialization techniques can be categorized in the following three groups: (1) randomness, an initialization method which generates different individuals using a pseudo-random number generator (PRNG) [19]which, if it uses different initial seeds (those fed into the generator) and produces different individuals, is considered *stochastic* whereas, if it produces the same individuals regardless of any initial seed, such as the chaotic number generator (CNG) [20], is considered *deterministic*; (2) compositionality whereby all basic techniques which generate initial individuals in only one step are considered *non-compositional* and those consisting of more than one, such as the opposition-based learning (OBL) method, *compositional* [21]; and (3) generality which consists of either generic or application-specific techniques, with the former meaning that an initialization mechanism can be directly used for all problems while, in the latter, a technique is specially designed for particular real-world problems, i.e., timetabling [22].

However, existing approaches suffer from some drawbacks; for instance, none considers any information from the behavior of the function to be optimized despite the fact that such information may provide important clues to identifying the approximate regions of interest for exploration. It is worth noting that exploring regions of the search space that do not contain useful solutions may consume unnecessary computational efforts. As shown in the plots provided in Section IV,

---

[1] The authors are with the School of Engineering and Information Technology, University of New South Wales Canberra, Australia, emails: s.elsayed@adfa.edu.au, r.sarker@adfa.edu.au.

[2] The author is with the Depto. de Computación, CINVESTAV-IPN, Mexico, email: ccoello@cs.cinvestav.mx.

for different functions, existing initialization methods provide either very little or no useful information about the function's behavior. Although it is common trying to generate an initial population by covering the whole search space, the individuals generated by many approaches do not uniformly cover the search space,. In this case, the evolutionary search process may not find good-quality solutions in an efficient way. Although uniform experimental and orthogonal designs [9] have the capability to tackle this drawback, they do not provide any indication of the function's behavior that may help to generate a good mix of individuals for an effective and efficient evolution.

Motivated by the abovementioned research gap, in this paper, a new initialization technique that generates individuals by covering the entire search space is proposed. In this process, the entire search space is divided into a number of unit spaces in a given sequence within the variables' bounds. Individuals are then generated by taking the corner points of those unit spaces. When the fitness values of these individuals are considered in the sequence in which they are generated, they may provide interesting behaviour, as shown in Section IV. This type of analysis can help us choose an initial population from the search space with non-uniform discrete points that would positively influence the optimization process. To the best of our knowledge, no such technique has been proposed in the literature. It is worth mentioning that this method is different from an orthogonal design as it does not require the construction of orthogonal arrays, and from Latin hypercube sampling (LHS) in the way points are generated and their sequence, as is clarified later.

Firstly, a set of 13 well-known test problems was chosen to analyze the functions' behaviors using the proposed method and judge the influence of the initial population generated on the performance of a well-known DE algorithm. This approach was then tested using several state-of-the-art DE algorithms to solve another set of well-known 30 unconstrained problems. For the first set of test problems, the proposed method provided interesting insights about the functions' behaviors that cannot be obtained using the other four popular initialization methods considered in this paper. In addition, the initial population generated improved the algorithm's performance by 43%. A similar analysis was conducted using five well-known DE algorithms to solve the second set of test problems, with the proposed method able to significantly improve their performances.

This method was also tested with other CI approaches, namely, GA, CMA-ES and two variants of PSO, with the results demonstrating its benefits for improving the performances of those algorithms. Furthermore, the results obtained from a well-known multi-objective technique for solving three well-known multi-objective problems showed that this algorithm was able to achieve better performance using the proposed method.

The remainder of this paper is organized as follows. An overview of different initialization approaches and a brief review of DE are presented in Section II. Our proposed approach is then discussed in Section III. The experimental results and conclusions are discussed in Sections IV and V, respectively.

II. REVIEW

In this section, a brief review of different initialization techniques and DE is provided.

A. Initialization Methods

The most commonly used technique for initializing a population of individuals in an EA is the PRNG which generates a sequence of numbers the characteristics of which approximate those of sequences of random numbers [19], that is, the points are scattered according to a statistical distribution, such as the uniform one described in equation 5. In fact, it is quite difficult to ensure that the sequence generated by the PRNG is truly random, as it is determined by a small set of initial values (the PRNG's seed) [23]. Although this mechanism is simple, it suffers from the curse of dimensionality [24] and cannot generate properly distributed points [19], [25].

The CNG, which is based on the chaos theory [26] and studies the behaviors of dynamical systems, has been adopted with EAs [20]. Mathematically speaking, chaos is the randomness of a simple deterministic dynamical system, and a chaotic system may be considered a source of randomness [20]. To generate a chaotic sequence, a chaotic mapping is needed in which, in its simplest form, a variable can be generated as

$$x_{i,j}^{t} = f_{ch}\left(x_{i,j}^{t}\right) \tag{2}$$

where f_{ch} is the chaotic mapping and $x_{i,j}^{t}$ is the j^{th} variable of the i^{th} individual at generation t. It is worth mentioning that there are several mapping functions, such as logistic, circle, sinus and tent, in the literature [20]. Such maps generate real-values $\in [0,1]$ which can be used in equation (5), by replacing $rand_j(0,1)$, to initialize any individual in the population. Of seven different chaotic maps adopted with DE, the variant with the sinus map has shown the best performance [20], better than that of DE with a uniform initialization [19].

Uniform experimental design (UED) [27] is a kind of space-filling mechanism which searches for solutions to be uniformly scattered in a given range. It defines the uniform array as $U_M q^D$, where Q is the number of levels and M the number of combinations randomly selected from Q^D combinations. However, as evaluating such a large population is practically impossible, even for small-scale problems, one possible way of overcoming this drawback is to use an orthogonal design.

The idea behind an orthogonal array is to specify a set of combinations scattered uniformly over the space of all possible combinations. As an orthogonal design is applicable to discrete factors/variables, Leung et al. [9] proposed a modified variant to improve the performance of GAs. In their proposed method, the domain of each variable $\left(\left[\underline{x}_j, \overline{x}_j\right]\right)$ was quantized into Q_j values, i.e., $\left\{\alpha_{j,1}, \alpha_{j,2}, ..., \alpha_{j,k}, ...\alpha_{j,Q_j}\right\}$, where Q_j was odd and $\alpha_{j,k},\ \forall\, k = \{1,2,...,Q_j\}$ the k^{th} level of the j^{th} variable as

$$\alpha_{j,k} = \begin{cases} \underline{x}_j & j = 1 \\ \underline{x}_j + (k-1)\left(\frac{\overline{x}_j - \underline{x}_j}{Q_j - 1}\right) & 2 \le k \le Q_j - 1 \\ \overline{x}_j & k = Q_j \end{cases} \tag{3}$$

Then, the smallest positive integer number J_j was calculated by fulfilling $\frac{Q_j^J - 1}{Q_j - 1} \ge D$. If $\frac{Q_j^J - 1}{Q_j - 1}$ was not exactly equal to D, D' would be set equal to D, otherwise assigned a value of $\lfloor \frac{Q_j^J - 1}{Q_j - 1} \rfloor$. Subsequently, the orthogonal array $\left(L_{Q_j^{J_j}}(Q_j^{D'})\right)$ was constructed as described in Algorithm 1 and then, the last $(D' - D)$ columns of $L_{Q_j^{J_j}}(Q_j^{D'})$ deleted, with this process undertaken for all variables, i.e., $j = \{1, 2, ..D\}$.

Algorithm 1 Construction of the orthogonal arrays

```
for $r = 1$ to $J$ // construct the basic columns do
    $j = \frac{Q_j^{r-1}-1}{Q_j-1} + 1$;
    for $i = 1$ to $Q_j^J$ do
        $a_{i,j} = \lfloor \frac{i-1}{Q_j^{J-r}} \rfloor \bmod Q_j$
    end for
end for
for $r = 2$ to $J$ // Construct the nonbasic columns do
    $j = \frac{Q_j^{r-1}-1}{Q_j-1} + 1$;
    for $s = 2$ to $j - 1$ do
        for $t = 1$ to $Q_j - 1$ do
            $a_{j+(s-1)(Q_j-1)+t} = (a_s \times t + a_j) \mathrm{mod} Q_j$;
        end for
    end for
end for
Increment $a_{i,j}$ by one $\forall\ 1 \leq i \leq M$ and $1 \leq j \leq D$.
```

In the literature, orthogonal design initialization has been successfully adopted with different CI approaches, such as DE [28] and PSO [29], and found to improve performances.

LHS [30] is another kind of space-filling mechanism that has been successfully adopted with EAs [31], [32]. It creates the initial grid by dividing the variables into a fixed number of intervals and then generates a random value within each interval so that there is only one design in each row or column. As explained later, the similarity between the proposed method and a LHS design is that they both divide the search space into grids. However, their two main significant differences are that, in the proposed method (1) the points generated are those at the corner points of grids with a possibility that more than one point can be generated in more than one grid in the same row or column; and (2) the generated points are represented in a two-dimensional plot (the number of generated points according to the sequence of generations versus the fitness values) which provides interesting information about the behavior of the function to be optimized that a LHS design does not.

Another initialization method that has demonstrated success over recent years is opposition-based learning (OBL) [21]. It can be seen as a composite method in which an initial population (original) of individuals is generated using any of the above methods and then a heuristic rule used to produce an opposite population $(\widetilde{x})$ of the original one, as

$$\widetilde{x}_{i,j} = \underline{x}_j + \overline{x}_j - x_{i,j},\ \forall i = \{1, 2, ..., PS\},\ D = \{1, 2, ..., D\} \tag{4}$$

Finally, a subset of the best individuals from $(x \cup \tilde{x})$ is selected. Many variations of this method have been proposed including center-based sampling [33], generalized OBL [34] and current optimum OBL [35], [36]. Although OBL methods perform well for solving many optimization problems, they have the three drawbacks that [19]: (1) they consume a part of the computational budget to evaluate the fitness functions and select the best subset of both populations; (2) as the secondary points are based on the original population, their performances depend on the quality of the original population to some extent; and (3) it is most likely that solutions which have useful information are discarded due only to their low fitness values.

However, from our perspective, the first drawback can be ignored when the initial population can capture useful information about the problem's characteristics which can significantly speed up the convergence rate. This is one of the main motivations for the work reported in this paper.

B. DE

DE is a population-based stochastic algorithm for global optimization [37]. Initially, a population of random D$-$dimensional vectors is generated, i.e., $X = \{\overrightarrow{x}_1, \overrightarrow{x}_2, ...\overrightarrow{x}_{PS}\}$, where PS is the population size. These individuals are then used to generate a mutant population of size PS, i.e., $V = \{\overrightarrow{v}_1, \overrightarrow{v}_2, ...\overrightarrow{v}_{PS}\}$. Then, each solution $(\overrightarrow{x}_i)$, in the current population, is recombined with its corresponding mutant vector $(\overrightarrow{v}_i)$ to generate a trial vector $(\overrightarrow{u}_i)$, where $i = 1, 2, ...PS$, and every $\overrightarrow{x}_i$ and its corresponding $\overrightarrow{u}_i$ are pair-wise compared, with the winning vectors becoming the parent population (X) in the next generation [1], [38]. Below is a brief description of each step in DE.

- **Initialization:** each individual in the population is represented as a D-dimensional vector and each variable is generated within its allowable boundaries:

  $$x_{i,j} = \underline{x}_j + rand_j(0,1) \times (\overline{x}_j - \underline{x}_j)\ \forall j = \{1, 2, ...D\} \tag{5}$$

  where $rand_j(0,1)$ is a uniform random number within $[0,1]$. Note that DE was initially proposed to solve only continuous problems and was later adapted to solve other types of optimization problems [1].
- **Mutation:** DE generates new solutions that are perturbations of the current ones in which, in its simplest form (DE/rand/1), a mutant vector $(\overrightarrow{v}_i)$ is generated by adding a scaled difference between two random solution vectors to a third one (equation (6)).

  $$\overrightarrow{v}_i = \overrightarrow{x}_{r_1} + F \times (\overrightarrow{x}_{r_2} - \overrightarrow{x}_{r_3}) \tag{6}$$

  where $\overrightarrow{x}_{r_1}$, $\overrightarrow{x}_{r_2}$ and $\overrightarrow{x}_{r_3}$ are distinct solution vectors in the current population and none is similar to $\overrightarrow{x}_i$, F a positive real number that controls the rate at which the population evolves [1]. $\overrightarrow{x}_{r_1}$ is also called the *base vector.*

  There are many variants of this operator, such as DE/best/1 [5], DE/rand-to-best/1 [39] and DE/current-to-best [40]. For more details, readers are referred to [41].
- **Crossover:** there are two well-known crossover schemes for DE: binomial and exponential. The former (sometimes also called uniform or discrete crossover [1]) is conducted on every $j \in [1, D]$ with a predefined crossover probability. In particular, for each j, a uniform random number $(rand_j(0,1))$ is generated. If its value is less than Cr, the trial value $(\overrightarrow{u}_{i,j})$ is copied from the corresponding value from the mutant vector $(\overrightarrow{v}_{i,j})$; otherwise, it is copied from the parent vector $(\overrightarrow{x}_{i,j})$ that is:

  $$u_{i,j} = \begin{cases} v_{i,j} & if\ (rand_j(0,1) \leq cr \,\text{or}\, j = j_{rand}) \\ x_{i,j} & \text{otherwise} \end{cases} \tag{7}$$

  $j_{rand} \in \{1, 2, ..., D\}$ is a randomly integer index which ensures that $\overrightarrow{u_i}$ obtains at least one component from $\overrightarrow{v_i}$. On the other hand, an exponential crossover is similar to a two-point crossover in which the first cutting point

(l) is randomly selected from the range $[1, D]$ and the second is determined such that L components are copied from $\overrightarrow{v}_i$ [42], as follows:

$$u_{i,j} = \begin{cases} v_{i,j} & \forall j = \langle l \rangle_D, \langle l+1 \rangle_D, ..., \langle l+L-1 \rangle_D \\ x_{i,j} & \forall j \in [1, D] \end{cases} \tag{8}$$

where $\langle l \rangle_D$ denotes a modulo function with a modulus of D and $L \in [1, D]$.

- **Selection:** DE uses a simple one-to-one survivor selection in which, at generation (t), a trial vector $(\overrightarrow{u}_{i,t})$ competes against the target/parent vector $(\overrightarrow{x}_{i,t})$, and the best from them, in terms of the fitness value and/or constraint violation, is considered a vector in the new population in the next generation $(t+1)$.

III. PROPOSED APPROACH

As previously discussed, one objective of this paper is to propose an initialization method which can cover a reasonable search space and can directly give an indication of the function's behavior. By achieving this, it can be possible to determine the search areas of interest that an EA can focus on. The details of the proposed method are discussed below.

Initially, the search domain $\left[\underline{\overrightarrow{x}} - \overline{\overrightarrow{x}}\right]$ is divided into $q+1$ segment vectors, where the first and last segment vectors are the lower and upper range vectors of all the decision variables, respectively. To do this, the interval (I) of each segment vector can be determined using equation (9).

$$\overrightarrow{I} = \frac{\left(\overline{\overrightarrow{x}} - \underline{\overrightarrow{x}}\right)}{q} \tag{9}$$

Then, the k^{th} segment vector $\left(\overrightarrow{S}_k,\ \forall k = \{1, 2, ..., q+1\}\right)$ is generated as follows:

$$\overrightarrow{S}_k = \begin{cases} \underline{\overrightarrow{x}} & k = 1 \\ \overrightarrow{S}_{k-1} + \overrightarrow{I} & 2 \le k \le q \\ \overline{\overrightarrow{x}} & k = q+1 \end{cases} \tag{10}$$

Example 1. Let's assume that there are 3 decision variables $(D = 3)$ and that the lower and upper limits of each of them are 0 and 1, respectively, i.e., $\underline{\overrightarrow{x}} = [0, 0, 0]$ and $\overline{\overrightarrow{x}} = [1, 1, 1]$. For $q = 4$,

$$\overrightarrow{I} = \frac{[1,1,1] - [0,0,0]}{4} = [0.25, 0.25, 0.25]\,,$$

hence $5\,(q+1)$ segment vectors can be generated as follows:

$$\begin{aligned} \overrightarrow{S}_1 &= \underline{\overrightarrow{x}} = [0.0, 0.0, 0.0] \\ \overrightarrow{S}_2 &= \overrightarrow{S}_1 + \overrightarrow{I} = [0.25, 0.25, 0.25] \\ \overrightarrow{S}_3 &= \overrightarrow{S}_2 + \overrightarrow{I} = [0.5, 0.5, 0.5] \\ \overrightarrow{S}_4 &= \overrightarrow{S}_3 + \overrightarrow{I} = [0.75, 0.75, 0.75] \\ \overrightarrow{S}_5 &= \overline{\overrightarrow{x}} = [1.0, 1.0, 1.0] \end{aligned}$$

Subsequently, a set of sequences can be generated as described in Algorithm 2. For the k^{th} vector in S $\left(\overrightarrow{S}_k\right)$, some possible points are generated, such that starting with the last variable, i.e., $j = D$, by changing its value to all possible $S_{k,j}\ \forall\ k = \{1, 2, ...q+1\}$. Note that, every change means a new point is generated. Also for every change, only the j^{th} variable is changing, while the remaining variables are set as those of $\overrightarrow{S}_k$. At the same time, no redundant points should be generated. It is worth mentioning that the order in which the points are generated is very important.

Algorithm 2 Population Initialization

```
define $\overrightarrow{\Re} = [0, 0, ..., 0]_{q+1}$ //used to avoid generating redundant points;
$i \leftarrow 0$; //to count a point to generate
for $k = 1$:$q+1$ do
    for $j = D : -1 : 1$ // for each decision variable) do
        for $r = 1$:$q+1$ do
            if $(k == r)$ and $(\Re(k) == 0)$ then
                $\Re(k) \leftarrow 1$;
                $i \leftarrow i + 1$;
                $\overrightarrow{x}_i \leftarrow \overrightarrow{S}_k$;
                $x_{i,j} \leftarrow S_{r,j}$; // change the $j^{th}$ variable in the $i^{th}$ point
            else if $(k == r)$ and $(\Re(k) == 1)$ then
                $r \leftarrow r + 1$; // skip generating a redundant point
            else if $k \sim= r$ then
                $i \leftarrow i + 1$;
                $\overrightarrow{x}_i \leftarrow \overrightarrow{S}_k$;
                $x_{i,j} \leftarrow S_{r,j}$; // change the $j^{th}$ variable in the $i^{th}$ point
            end if
        end for
    end for
end for
```

Based on this method, the maximum number of individuals that can be generated is:

$$(q+1) \times ((D \times q) + 1) \tag{11}$$

which is less than the maximum number of grids D^q, assumed by the UED initialization method. For example, for $D = 10$ and $q = 10$, the proposed method generates $(10+1) \times ((10 \times 10) + 1) = 1111$ points. This number is significantly less that of UED, which is 10^{10}.

Example 2. Let's assume that $D = 3$, and the lower and upper limits of each of them are 1 and 3, respectively, i.e., $\underline{\overrightarrow{x}} = [1, 1, 1]$ and $\overline{\overrightarrow{x}} = [3, 3, 3]$, and $q = 2$. As previously described in Example 1, S will have $q+1 = 3$ vectors, such as:

$$S = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$$

In reference to equation (11), 21 points can be generated in the following order:

$$\begin{array}{llllll} \overrightarrow{x}_1 = & [1,1,1]\,; & \overrightarrow{x}_8 = & [2,2,1]\,; & \overrightarrow{x}_{15} = & [3,3,1]\,; \\ \overrightarrow{x}_2 = & [1,1,2]\,; & \overrightarrow{x}_9 = & [2,2,2]\,; & \overrightarrow{x}_{16} = & [3,3,2]\,; \\ \overrightarrow{x}_3 = & [1,1,3]\,; & \overrightarrow{x}_{10} = & [2,2,3]\,; & \overrightarrow{x}_{17} = & [3,3,3]\,; \\ \overrightarrow{x}_4 = & [1,2,1]\,; & \overrightarrow{x}_{11} = & [2,1,2]\,; & \overrightarrow{x}_{18} = & [3,1,3]\,; \\ \overrightarrow{S}_5 = & [1,3,1]\,; & \overrightarrow{x}_{12} = & [2,3,2]\,; & \overrightarrow{x}_{19} = & [3,2,3]\,; \\ \overrightarrow{x}_6 = & [2,1,1]\,; & \overrightarrow{x}_{13} = & [1,2,2]\,; & \overrightarrow{x}_{20} = & [1,3,3]\,; \\ \overrightarrow{x}_7 = & [3,1,1]\,; & \overrightarrow{x}_{14} = & [3,2,2]\,; & \overrightarrow{x}_{21} = & [2,3,3]\,; \end{array}$$

IV. EXPERIMENTAL RESULTS

In this section, the results obtained for two different unconstrained data sets, and a few multi-objective problems, are presented and discussed.

A. Proposed initialization and function's behavior

To judge the capability of the proposed initialization technique to provide useful information about the behavioral pattern of the objective function, 13 test problems were considered. Their mathematical characteristics are provided in Table 1 in Appendix A where F_1 to F_7 are unimodal problems and F_8 to F_{13} multi-modal ones.

For all the problems, 10 and 30 variables were used. An initial population was generated using the proposed initialization method and the fitness values of all the individuals calculated, plotted and then compared with those of the (1) uniform, (2) orthogonal-based, (3) OBL based on a uniform sampling (OBL-uniform), and (4) LHS initialization methods. As a sample, we plotted only the 30D problems. For the proposed method, q was set to a value of 10, which means that PS was 1111 and 3311, for the 10D and 30D problems, respectively. Note that for the orthogonal-based initialization method, PS was 1000 and 3375, respectively. Also, for OBL-uniform, $\lceil \frac{PS}{2} \rceil$ individuals were generated using a uniform distribution. Then, the opposite points were generated to form PS. Also, the uniform and LHS methods started with the same population size as used in the proposed method, i.e., 1111 and 3311, for the 10D and 30D problems, respectively. Note that the fitness values were plotted according to the sequences in which the individuals were generated by the initialization methods.

Considering the unimodal problems (F_1 to F_7), the fitness values of the initial population given in figures 1 and 2 in the supplementary file, with an example for F_{05} shown in Figure 1, reveal the capability of the proposed method to indicate that F_1 to F_7 were unimodal, with the best minimum value close to zero. In contrast, all the other methods did not directly provide such important information.

For the multi-modal problems, the fitness values of the initial population given in figures 3 to 5 in the supplementary file, with two examples shown in Figure 1. It was found that the proposed method was able to determine the multi-modality of the landscapes of F_8 and F_{10}, while for the other 4, although the areas of interest were around the center of the search domain, it was still not clear that the problems were multi-modal. Therefore, we increased q to 20 and plotted all the fitness values (the 3rd plot in each sub-figure), which clearly showed that the problems are multi-modal. For F_{11} to F_{13}, it was evident that the global solutions were around the center of the search domain and, by zooming in (i.e., generating more points within a smaller search domain around the center of the search space, i.e., 10% of the search domain), the problems appeared multi-modal.

Concerning the quality of the initial solutions obtained, the best fitness value in the initial population of each method and the average fitness values from 25 runs recorded are presented in Table I. Note that, in these experiments, the proposed method used a q value of 10 and its results revealed that it was able to obtain much better results for all problems, except F_8 with 30D for which the orthogonal-based method was slightly better. For some test problems the optimal solutions were obtained within the first initialization. It was also noted that the performances of the uniform and OBL initialization methods came in the 2nd and 3rd places, respectively, and, interestingly, LHS was the worst.

Table II. Ranks of JADE with proposed, uniform, OBL-uniform, LHS, and orthogonal initialization methods based on Friedman's Test

Criteria	**JADE**				
	Proposed	Uniform	OBL-uniform	LHS	Orthogonal
Best fitness values	**2.27**	3.19	3.12	3.58	2.85
Average fitness values	**2.00**	3.38	2.69	3.23	3.69

Furthermore, all initialization methods were implemented with a well-known DE and well-performing algorithm, known as JADE [40], and run 25 times to solve all the problems with $D = 30$. For a fair comparison, the PS was initially set to a value of 3311, except for the orthogonal-based one which was set to 3375, and then the best 100 individuals selected to form the initial population. As selecting the best solutions from a large population of individuals may reduce diversity, especially for multi-modal problems, and to gain benefits from the proposed method, the selected individuals could be taken from different areas of the search space in which we were interested; for instance, a_1 to a_5 in Figure 2 for F_8. Subsequently, we placed more emphasis on the best area (a_1) by taking the best 60 individuals from it and the remaining 40 solutions from all the other areas (the best 10 in each). Note that, although this selection procedure was undertaken for all the multi-modal problems (F_8 to F_{13}), any other one could be used depending on the problem considered (as discussed in Algorithm 3). Also, it was noted that, in Figure 2, a jump in the fitness values occurred every 301 solutions. The detailed results shown in Table 2 in Appendix A indicated that JADE with the proposed initialization method was the best.

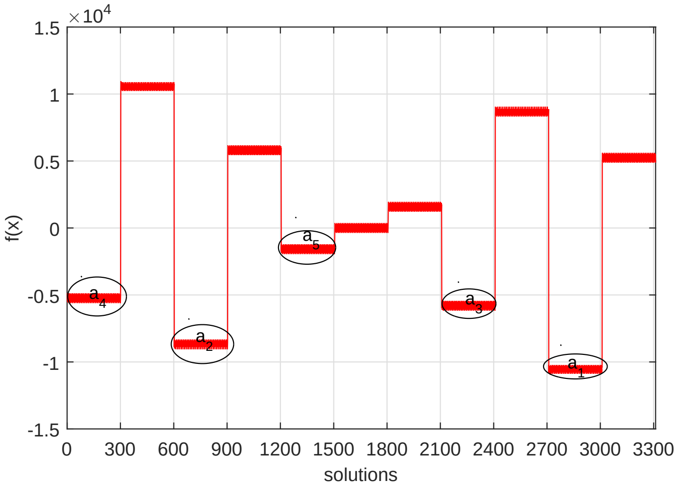
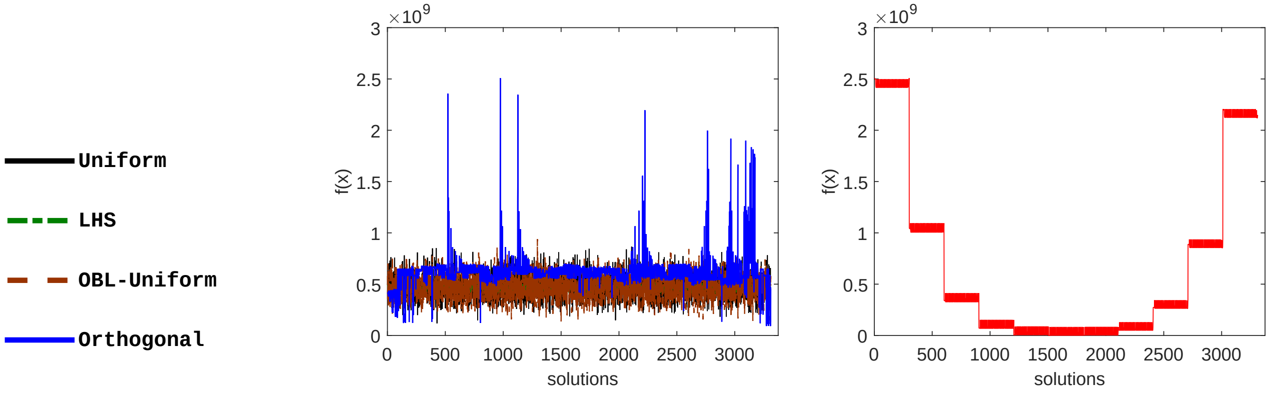


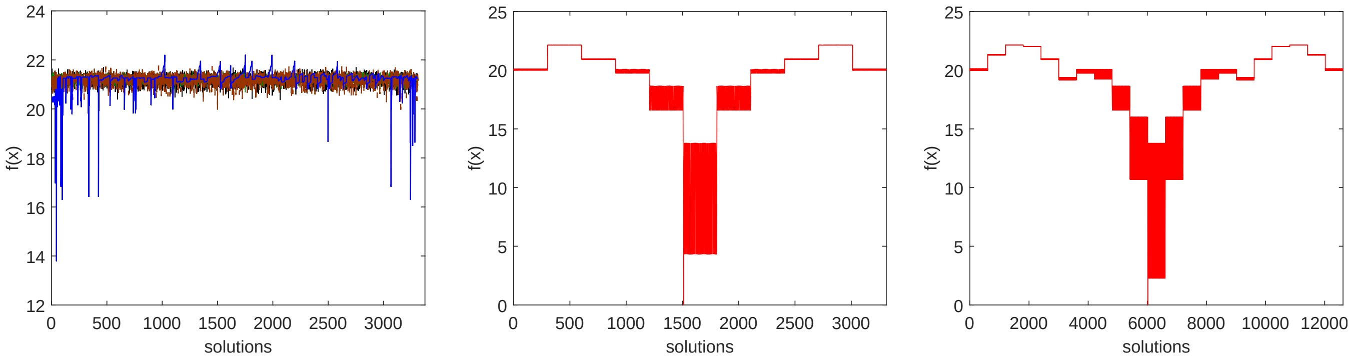
Figure 2. Search areas of interest for F_8

From a statistical perspective, the Friedman test was undertaken to rank all the variants based on the best and average fitness values obtained, with the results shown in Table II. Considering the best results, it was found that JADE with the proposed method was ranked first, followed by JADE with the orthogonal, OBL-uniform, uniform and LHS initialization methods, respectively. Regarding the average results, JADE with the proposed initialization method was ranked first, followed by JADE with the OBL-uniform, LHS, uniform and orthogonal methods, respectively.

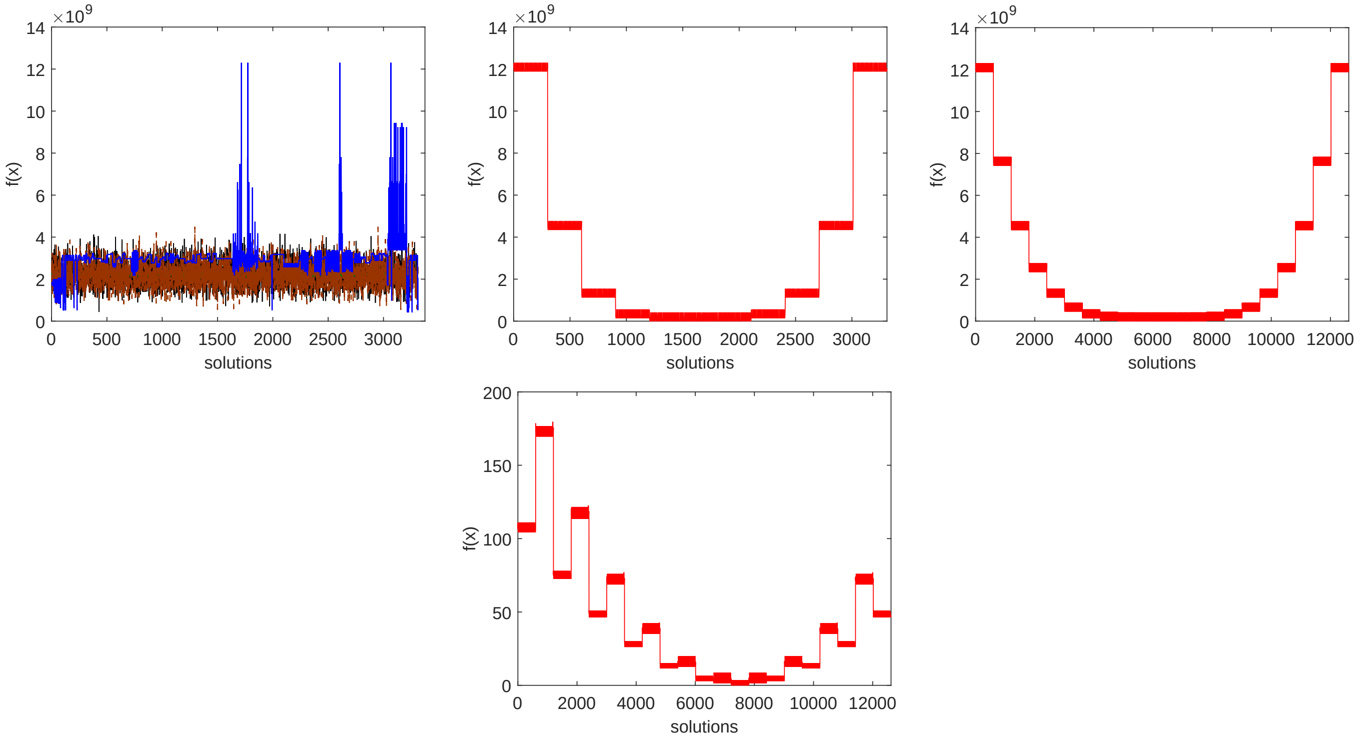
In addition, the average numbers of fitness evaluations consumed to reach optimal solutions with a tolerance of $1\text{E}-04$ were recorded, as shown in Table IIIin which it is clear that JADE with the proposed method was faster than all the other variants. To provide more information, the error bars plots are depicted in Figure 3, with the curve represent the average



(a) F_5



(b) F_{10}



(c) F_{13}

Figure 1. Fitness values of the initial population using different initialization methods for F_1, F_9, and F_{13} with 30D (the plots in red (2^{nd} to 4^{th} plots in each sub-figure) correspond to the proposed method, where the 2^{nd} and 3^{rd} plots, in each sub-figure, are with $q = 10$ and 20, respectively, while the 4^{th} plot in F_{13} is for 10% of the search domain, i.e., $[0.1\underline{x}, 0.1\bar{x}]$)

Table I. AVERAGE OF BEST SOLUTIONS OBTAINED WITHIN THE INITIAL POPULATION, OUT OF 25 RUNS, USING DIFFERENT INITIALIZATION METHODS

Prob.	D	Optimal	Proposed	Uniform	OBL-uniform	LHS	Orthogonal
F_1	10D	0.0000E+00	**0.0000E+00**	7.5401E+03	8.3748E+03	2.7048E+04	2.0000E+04
	30D		**0.0000E+00**	4.7089E+04	4.8362E+04	9.5737E+04	4.0408E+04
F_2	10D	0.0000E+00	**0.0000E+00**	7.5401E+03	8.3748E+03	2.7048E+04	1.6941E+05
	30D		**0.0000E+00**	5.6275E+07	1.0130E+08	1.6420E+13	1.3000E+02
F_3	10D	0.0000E+00	**0.0000E+00**	8.1260E+03	8.2997E+03	1.0787E+04	4.9012E+04
	30D		**0.0000E+00**	6.0228E+04	6.4820E+04	5.6530E+04	2.2367E+05
F_4	10D	0.0000E+00	**0.0000E+00**	4.6789E+01	5.1335E+01	8.0503E+01	5.5556E+01
	30D		**0.0000E+00**	7.4159E+01	7.5353E+01	9.3419E+01	1.0000E+02
F_5	10D	0.0000E+00	**9.0000E+00**	7.2785E+06	9.2222E+06	7.0021E+07	8.1769E+07
	30D		**2.9000E+01**	1.2456E+08	1.3322E+08	3.8244E+08	1.7222E+08
F_6	10D	0.0000E+00	**0.0000E+00**	7.5331E+03	8.3870E+03	2.7034E+04	1.9801E+04
	30D		**0.0000E+00**	4.7110E+04	4.8321E+04	9.5687E+04	4.0479E+04
F_7	10D	0.0000E+00	**5.2957E-02**	2.1332E+00	2.3438E+00	1.1862E+01	4.5446E+00
	30D		**6.2630E-02**	5.8535E+01	6.4948E+01	1.4085E+02	1.6634E+01
F_8	10D	-4.1898E+03	**-3.6518E+03**	-2.0411E+03	-1.9911E+03	-1.5707E+03	-2.8573E+03
	30D	-1.2569E+04	-1.0955E+04	-3.7965E+03	-3.7388E+03	-1.0429E+03	**-1.2120E+04**
F_9	10D	0.0000E+00	**0.0000E+00**	8.5474E+01	8.7250E+01	1.1805E+02	1.4142E+02
	30D		**0.0000E+00**	3.6385E+02	3.7220E+02	4.8060E+02	1.9271E+02
F_{10}	10D	0.0000E+00	**7.9936E-15**	1.8089E+01	1.8353E+01	2.0207E+01	1.9875E+01
	30D		**7.9936E-15**	2.0007E+01	2.0107E+01	2.0643E+01	1.3783E+01
F_{11}	10D	0.0000E+00	**0.0000E+00**	6.8859E+01	7.6370E+01	2.4443E+02	1.8100E+02
	30D		**0.0000E+00**	4.2481E+02	4.3626E+02	8.6263E+02	3.6467E+02
F_{12}	10D	0.0000E+00	**2.6507E+00**	7.2630E+07	1.2228E+08	2.2617E+09	2.5600E+09
	30D		**1.6690E+00**	2.2131E+09	2.5142E+09	1.0782E+10	3.0547E+09
F_{13}	10D	0.0000E+00	**1.0000E+00**	2.7971E+07	3.8337E+07	4.4821E+08	4.1006E+08
	30D		**3.0000E+00**	5.0110E+08	5.5257E+08	1.9837E+09	6.2132E+08

Table III. AVERAGE FITNESS EVALUATIONS TO REACH $\left|f\left(\overrightarrow{x}_{best}\right)-f\left(\overrightarrow{x^*}\right)\right| \leq 1e-04$ WITH DIFFERENT INITIALIZATION METHODS FOR 30D PROBLEMS

Prob.	Best				
	Proposed	Uniform	OBL-uniform	LHS	Orthogonal
F_1	**3.3110E+03**	2.3875E+04	2.4012E+04	2.4563E+04	2.4399E+04
F_2	**3.3110E+03**	3.6359E+04	3.6496E+04	3.8243E+04	3.6863E+04
F_3	**3.3110E+03**	5.3839E+04	5.1444E+04	5.1451E+04	5.2363E+04
F_4	**3.3110E+03**	5.0155E+04	4.9776E+04	5.0579E+04	5.1579E+04
F_5	**9.6871E+04**	1.0334E+05	1.0650E+05	9.9783E+04	1.0486E+05
F_6	**3.3110E+03**	1.4439E+04	1.4580E+04	1.5283E+04	1.5059E+04
F_7	**3.0000E+05**	1.4439E+04	1.4580E+04	1.5283E+04	1.5059E+04
F_8	**4.8667E+04**	9.9491E+04	9.9864E+04	1.0202E+05	9.6595E+04
F_9	**3.3110E+03**	1.0468E+05	1.0442E+05	1.0524E+05	1.0557E+05
F_{10}	**3.4110E+03**	3.1427E+04	3.1388E+04	3.1867E+04	4.0807E+04
F_{11}	**3.3110E+03**	2.6375E+04	2.7444E+04	2.7567E+04	2.6687E+04
F_{12}	**1.5095E+04**	2.1443E+04	2.1708E+04	2.2331E+04	2.1631E+04
F_{13}	**1.9155E+04**	2.3811E+04	2.3852E+04	2.4443E+04	2.3895E+04

Table IV. AVERAGE COMPUTATIONAL TIME, IN SECONDS, FOR DIFFERENT INITIALIZATION METHODS OVER13 TEST PROBLEMS WITH 30D

	Proposed	Uniform	OBL-uniform	LHS	Orthogonal
For initialization	0.00147	0.00111	**0.00082**	0.24711	0.01026
After optimization	**1.17**	2.57	2.63	2.83	2.54

fitness values over 25 runs for all generations. These figures clearly indicated the effectiveness of the proposed method and its capability to converge faster than the other variants. It was interesting to learn that even the orthogonal-based technique started with a better initial points then the proposed one when solving F_{08}, while JADE with the proposed method was able to converge to the optimal solution first.

1) Computational times: In this subsection, the computational times of all the methods are compared.

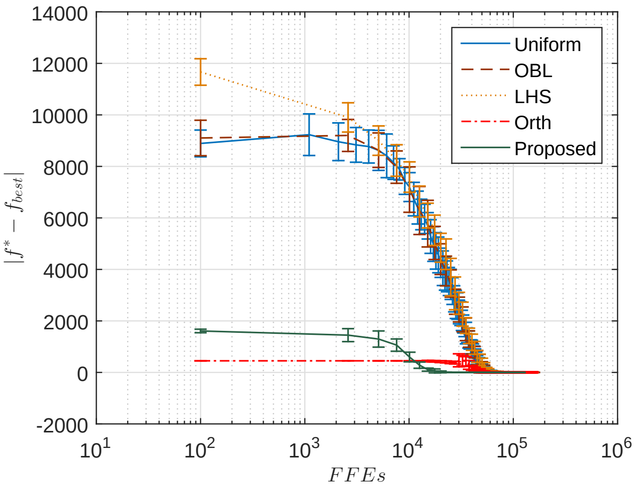
For each method, the average computational times taken to generate an initial population of 3311 individuals, i.e., $q = 10$, for the 13 test problems were calculated. Then, JADE was run 25 times to solve each of the 30D problems, with their average computational times recorded if one of the following two criteria was met: (1) the maximum number of fitness evaluations was reached; or (2) $\left|f\left(\overrightarrow{x}_{best}\right)-f\left(\overrightarrow{x^*}\right)\right| \leq 1e-04$. Note that all experiments were run on a PC with a Core(TM) i7-3770 CPU @ 3.40GHz (8 CPUs), 16 GB RAM and Windows 7 using MATLAB 8.5.0.197613 (R2015a).

Based on the results presented in Table IV, to generate 3311 initial points, it was found that the OBL-uniform method was the fastest. This was expected as only half its population was uniformly generated with calculations of the opposite points taking less time than generating them randomly. The uniform initialization method was second, with a small time difference from the proposed method. Also, the proposed method was clearly faster than the orthogonal-based and LHS ones. On the other hand, the computational times after completion of the optimization process provided different rankings. JADE with the proposed method was significantly faster than all the other variants, slower with the OBL-uniform than with the orthogonal-based and uniform methods and slowest with LHS.

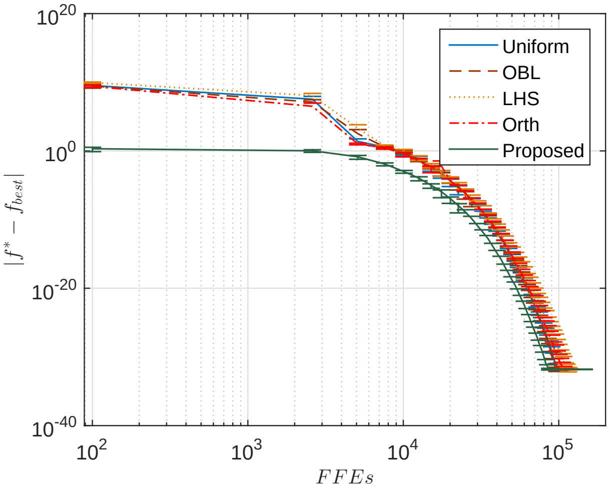
B. Solving CEC2014 test problems

In this section, five well-known DE algorithms are tested on the CEC2014 test suite [43], which contains 30 problems, from which F_1 to F_3 are unimodal functions, F_4 to F_{16} multi-modal, F_{17} to F_{22} hybrid and F_{23} to F_{30} compositions. The algorithms were (1) DE with self-adaptation of its control parameters (jDE) [44]; (2) DE with an ensemble of parameters and mutation strategies (EPSDE) [45]; (3) DE with composite trial vector generation strategies (CoDE) [46]; (4) success-history parameter adaptation of DE (SHADE) [47]; and (5) improved SHADE with linear population size reduction (LSHADE) [48]. For all these algorithms, the PS was set to a value of 100, except for LSHADE which used 18D individuals as recommended in the corresponding paper, with all the other parameter settings exactly the same as those in published papers.

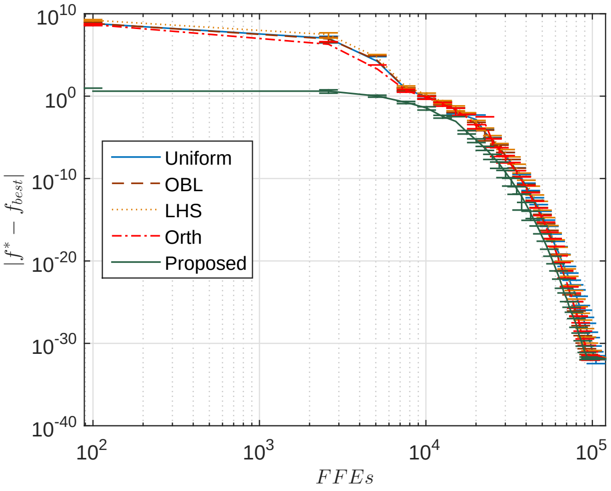
Each algorithm was run 51 times with the proposed initialization method using the uniform, OBL and LHS techniques and the stopping criterion $10,000D$ fitness evaluations, or $\left|f\left(\overrightarrow{x}_{best,t}\right)-f\left(\overrightarrow{x^*}\right)\right| \leq 1e-08$, where $f\left(\overrightarrow{x}_{best,t}\right)$ and



(a) F_8(x-axis is in log scale)



(b) F_{12} (x- and y-axes are in log scale)



(c) F_{13}(x- and y-axes are in log scale)

Figure 3. Convergence plots of JADE with all initialization methods for F_{08}, F_{12} and F_{13} with 30D

Algorithm 3 Generating initial points to use in solving the CEC2014 problems

```
With $q = 10$, generate an initial population $X = \{\overrightarrow{x}_1, \overrightarrow{x}_2, ..., \overrightarrow{x}_{PS}\}$, where $PS = 3311$, as described in Section III;
Calculate the fitness value of each individual ($f(\overrightarrow{x}) \forall\ i = 1, 2, ..., PS$)
Update the number of fitness evaluations to 3311;
Set $FFE_{max} \leftarrow$ 10000D - $\left(\frac{3311}{51}\right) \approx 299935$;
Divide $X$ into $n_g$ groups, i.e., $X = \{X_1, X_2, ..., X_{n_g}\}$, based on the problem on hand, each is of size $PS_g$;
Calculate the average fitness value of each group, i.e. $\bar{f}_g = \frac{\sum_{i=1}^{PS_g} f(\overrightarrow{x})}{PS_g}\ \forall g = 1, 2, ..n_g$;
$denominator \leftarrow \sum_{g=1}^{n_g} \frac{\sum_{gf=1}^{n_g} \bar{f}_{gf}}{f_g}$
Determine how many individuals to select from each $g$, i.e., $rate_g = \left(\left(\frac{\sum_{gf=1}^{n_g} \bar{f}_{gf}}{f_g}\right) \times denominator^{-1}\right), \forall g = 1, 2, ..n_g$;
$PS_g = \frac{rate_g}{\sum_{l=1}^{n_g} rate_l} \times \frac{PS}{2}, \forall g = 1, 2, ..n_g$
for $g = 1 : n_g$ do
    Insert the best individual in $X_g$;
    Insert random $PS_g - 1$ individuals in $X_g$;
end for
for $g = 1 : n_g$ do
    for $i = 1 : PS_g$ do
        Generate $\overrightarrow{\triangle}_i = \{\triangle_{i,1}, \triangle_{i,2}, ..., \triangle_{i,D}\}$, where $\triangle_{i,j} \in [-1, 1]$;
        $\overrightarrow{x}'_i \leftarrow \overrightarrow{x}_i + \overrightarrow{\triangle}_i$;
        Calculate $f'_i$;
        if $f\left(\overrightarrow{x}'_i\right) < f\left(\overrightarrow{x}_i\right)$ then
            $\overrightarrow{x}_i \leftarrow \overrightarrow{x}'_i$
            $f_i \leftarrow f'_i$
        end if
    end for
end for
Generate the remaining $\frac{PS}{2}$ using a uniform distribution.
```

$f\left(\overrightarrow{x^*}\right)$ are the fitness values of the best vector in generation t and optimal solution, respectively, considering only the 30D problems.It is worth mentioning that, although in the previous section, the initialization method was deterministic, starting with a different population in each run was a condition of the CEC2014 competition problems. Therefore, the procedure presented in Algorithm 3, was carried out to add randomness to the initial population generated by the proposed method, with a pool of individuals based on it with $q = 10$ generated. Then, as the corresponding fitness values were calculated, the number of fitness evaluations was 3311, a step that was performed only once. To provide a fair comparison, this number of fitness evaluations was deducted from the overall stopping criterion and distributed over 51 runs, so that it would be10000D - $\left(\frac{3311}{51}\right) \approx 299935$ fitness evaluations. Based on the shape of the curve of the fitness values of a problem considered, $\frac{PS}{2}$ individuals were selected, such that (1) all 3311 were divided into different groups (n_g), each of which was of size PS_g; (2) based on the average fitness values in each group, n_g solutions were selected, i.e., the best individual in each group was selected, while the remaining $PS_g - 1 \, \forall \, g = \{1, 2, ..., n_g\}$ were selected randomly at the beginning of each run. Then, for each individual $\overrightarrow{x}_i \, \forall i = \left\{1, 2, ..., \frac{PS}{2}\right\}$, a new shifted vector was generated, such that $\overrightarrow{x}'_i = \overrightarrow{x}_i + \overrightarrow{\triangle}_i$, where $\overrightarrow{\triangle}_i = \{\triangle_{i,1}, \triangle_{i,2}, ..., \triangle_{i,D}\}$ and $\triangle_{i,j} \in [-1, 1]$. Then, $\overrightarrow{x}'_i$ would replace $\overrightarrow{x}_i$ if $f\left(\overrightarrow{x}'_i\right) < f\left(\overrightarrow{x}_i\right)$ and simultaneously update the number of fitness evaluations. The remaining $\frac{PS}{2}$ individuals were generated using a uniform distribution.

Firstly, based on the curve of the fitness values obtained for each problem, it was noted that the CEC2014 benchmark problems were more difficult and challenging than the 13 problems considered in the previous section. In other words, most of the problems had many local optima. Some of these plots are shown in Figure 6 in the supplementary materials.

More detailed results regarding the quality of the solutions obtained are shown in Appendix A with a comparison summary presented in Table V. From these results, It is clear that all the algorithms considered were improved by using the proposed initialization method but, for a few multi-modal and hybrid functions, were inferior to those using other initialization approaches. One reason for this might have been that the initial population contained many solutions close to each other, a shortcoming that could be tackled by: (1) considering a different DE mutation operator which could maintain diversity; (2) changing the control parameters; or (3) increasing the size of the initial population by adding more diverse points. Broadly speaking, the proposed method could provide useful information to help select the proper parameter settings for solving a problem.

Also, the Wilcoxon signed rank test [49] was used to statistically compare the algorithms. Using a significance level of $p = 5\%$, one of three symbols ($+$, $-$, and $\approx$) was assigned, where $+$ means that the 1^{st} algorithm was statistically superior to the 2^{nd}, $-$ that the 1^{st} algorithm was statistically inferior to the 2^{nd} and $\approx$ that there was no significant difference between the two algorithms. The results in Table V show that each algorithm using the proposed initialization method was statically superior to its corresponding ones using other initialization mechanisms.

Also, based on the average fitness values obtained, the Friedman test was undertaken to rank all the variants of each DE algorithm, as shown in Table VIin which it is clear that the proposed method was ranked first. Of the uniform, OBL and LHS initialization methods, it was interesting to note that one technique might have been the best choice for one optimization algorithm but the worst for another; for instance, it was observed that using the uniform initialization mechanism with jDE, EPSDE and CODE was the worst option but with SHADE and LSHADE the best. Also, LHS with jDE was ranked 2nd but with CoDE and LSHADE was the worst.

C. *Testing proposed method with other CI methods*

In this section, the proposed method is tested on a few other algorithms : (1) GA with multi-parent crossover (GA-MPC) [50], (2) covariance matrix adaption-ES (CMA-ES) [6], (3) global and local PSO variants (g-PSO and l-PSO, respectively). Both g-PSO and l-PSO followed the variant proposed in [51]. Note that l-PSO used a typical ring topology, i.e., each particle interacted with only its immediate left and right neighbors. The parameter settings of all the algorithms are shown in Table VII.

Table V. Comparison summary among different algorithms with uniform, OBL, LHS and proposed initialization methods. The first algorithms in the first column use the proposed method.. Values in columns 4, 5 and 6 refer to numbers of test problems which an optimization algorithm with proposed initialization is better, similar and worse than the second algorithm in the 1st column, based on best and average fitness values obtained, respectively. p is a probability that is used to take a decision based on the Wilcoxon test

Algorithms	**Criteria**	**Better**	**Similar**	**Worse**	p
jDE vs. jDE(uniform)	Best results	18	6	6	0.016(+)
	Average results	20	5	5	0.001(+)
jDE vs. jDE(OBL)	Best results	19	6	5	0.092(+)
	Average results	19	4	7	0.005(+)
jDE vs. jDE(LHS)	Best results	17	6	7	0.049(+)
	Average results	18	4	8	0.009(+)
EPSDE vs. EPSDE(uniform)	Best results	16	6	8	0.097($\approx$)
	Average results	21	5	4	0.0004(+)
EPSDE vs. EPSDE(OBL)	Best results	17	6	7	0.03(+)
	Average results	18	5	7	0.005(+)
EPSDE vs. EPSDE(LHS)	Best results	16	6	8	0.034(+)
	Average results	19	4	7	0.004(+)
CoDE vs. CoDE(uniform)	Best results	25	0	5	0.0001(+)
	Average results	24	0	4	0.0000(+)
CoDE vs. CoDE(OBL)	Best results	24	0	6	0.0001(+)
	Average results	22	0	8	0.001(+)
CoDE vs. CoDE(LHS)	Best results	19	0	11	0.006(+)
	Average results	22	1	7	0.0002(+)
SHADE vs. SHADE(uniform)	Best results	21	7	2	0.0005(+)
	Average results	21	4	5	0.007(+)
SHADE vs. SHADE(OBL)	Best results	15	6	5	0.002(+)
	Average results	22	4	4	0.001(+)
SHADE vs. SHADE(LHS)	Best results	18	6	6	0.008(+)
	Average results	20	4	6	0.004(+)
LSHADE vs. LSHADE(uniform)	Best results	16	9	5	0.014(+)
	Average results	17	8	5	0.024(+)
LSHADE vs. LSHADE(OBL)	Best results	16	8	6	0.010(+)
	Average results	16	8	6	0.012(+)
LSHADE vs. LSHADE(LHS)	Best results	14	8	8	0.022(+)
	Average results	18	8	4	0.006(+)

Table VI. Ranks of all variants of jDE, EPSDE, CoDE, SHADE and LSHADE based on the Friedman Test

	Proposed	Uniform	OBL	LHS
jDE	**1.85**	2.87	2.78	2.5
EPSDE	**1.83**	2.97	2.45	2.75
CoDE	**1.65**	2.8	2.53	3.02
SHADE	**1.7**	2.45	2.95	2.9
LSHADE	**2.07**	2.42	2.53	2.98

Detailed test results are shown in Appendix A, with a comparison summary presented in Table VIII. It was found that all the algorithms were improved using the proposed initialization method. In addition, the Wilcoxon test was performed on all the problems and, based on the best and average results obtained, the GA-MPC, g-PSO and l-PSO algorithms using the proposed method were statistically superior to those using other initialization methods. Also, it was found that, when the proposed method was adopted with CMA-ES, there was a statistical difference based on the average fitness values but none considering the best ones.

Furthermore, the Friedman test was undertaken to rank each variant of the optimization algorithms used based on the average results it obtained. Table IX presents details of the rankings, with the results demonstrating the superiority of the proposed method.

Table VII. Configurations of GA-MPC, CMA-ES, g-PSO and l-PSO

Algorithms	Parameter settings
GA-MPC	$PS = 90$, $Cr = 1$ and $p = 0.1$ [50]
CMA-ES	$\mu = \frac{PS}{2}$, $\sigma = 0.3$, $PS = 100$ [12], [52]
g-PSO and l-PSO	$c_1 = c_2 = 1.49445$, $\chi = 0.729$ [51], and $PS = 100$

Table VIII. Comparison summary among GA-MPC, CMA-ES, g-PSO and l-PSO with different initialization techniques. The first algorithms in the first column use the proposed method. Values in columns 4, 5 and 6 refer to numbers of test problems which an optimization algorithm with proposed initialization is better, similar and worse than the second algorithm in the 1^{ST} column, based on best and average fitness values obtained, respectively. p is a probability that is used to take a decision based on the Wilcoxon test

Algorithms	**Criteria**	**Better**	**Similar**	**Worse**	p
GA-MPC vs. GA-MPC(uniform)	Best results	18	4	8	0.038(+)
	Average results	22	2	6	0.003(+)
GA-MPC vs.GA-MPC(OBL)	Best results	20	4	6	0.001(+)
	Average results	24	2	4	0.001(+)
GA-MPC vs. GA-MPC(LHS)	Best results	19	4	7	0.041(+)
	Average results	22	2	6	0.002(+)
CMA-ES vs. CMA-ES(uniform)	Best results	13	8	9	0.227(≈)
	Average results	20	4	6	0.012(+)
CMA-ES vs. CMA-ES(OBL)	Best results	14	8	8	0.236(≈)
	Average results	20	5	5	0.009(+)
CMA-ES vs. CMA-ES(LHS)	Best results	17	7	9	0.301(≈)
	Average results	21	4	5	0.001(+)
g-PSO vs. g-PSO(uniform)	Best results	22	0	8	0.026(+)
	Average results	26	0	4	0.000(+)
g-PSO vs. g-PSO(OBL)	Best results	22	0	8	0.028(+)
	Average results	25	0	5	0.000(+)
g-PSO vs. g-PSO(LHS)	Best results	23	0	7	0.022(+)
	Average results	26	0	4	0.000(+)
l-PSO vs. l-PSO(uniform)	Best results	25	1	4	0.000(+)
	Average results	25	0	5	0.001(+)
l-PSO vs. l-PSO(OBL)	Best results	21	1	8	0.005(+)
	Average results	25	0	5	0.001(+)
l-PSO vs. l-PSO(LHS)	Best results	23	1	6	0.004(+)
	Average results	23	0	7	0.003(+)

Table IX. Ranks of all variants of GA-MPC, CMA-ES, g-PSO and l-PSO based on the Friedman Test

	Proposed	Uniform	OBL	LHS
GA-MPC	**1.63**	2.5	3.2	2.67
CMA-ES	**1.75**	2.67	2.78	2.8
g-PSO	**1.43**	2.73	2.63	3.2
l-PSO	**1.57**	3.07	2.70	2.67

D. Solving multi-objective problems

In this section, solving three multi-objective problems using a well-known multi-objective EA based on decomposition (MOEA/D) [53] with the uniform and proposed initialization methods is discussed. The parameter settings of MOEA/D were the same as those reported in the abovementioned paper. Three well-known test problems (ZDT1, ZDT2 and ZDT3) with two objective functions were solved, where D was 30, the total number of runs 25 and maximum number of generations (g_{max}) 50.

To provide a fair comparison, 906, as $q = 5$, were initially generated then the best $PS = 101$ individuals selected based on the non-dominated sorting criterion [54]. Then, the hypervolume (HV) [55] indicator was calculated for the final Pareto front in each run. Note that the reference point was set at the maximum fitness value of each objective, among the non-dominated points, attained by both variants. Based on the HV results reported in Table X, it was found that MOEA/D was better with the proposed than uniform method. Furthermore, Figure 4shows the distribution of the final fitness values of the non-dominated solutions of the run with the best HV value of each algorithm for each test instance. It was evident that MOEA/D with the proposed initialization method was superior to that with the uniform random initialization one.

V. Conclusions and Future Work

Due to the crucial effect of the initial population on the performances of EAs, a considerable number of new initialization techniques has been proposed. However, they suffer from some drawbacks, such as (1) they are not able to directly indicate the pattern of the objective function, i.e., uni- or multi-modal; and (2) some do not uniformly cover the search space. Although some techniques try to tackle the second shortcoming, the first still exists. Therefore, in this paper, a new initialization technique was proposed. In it, the search domain of each decision variable was decomposed into a set of segments. Then, the search space was filled by systematically generating combinations of different segments of all the variables whereby the fitness values of the initial population were able to provide indications of the pattern of the function's behavior. As a consequence, based on the pattern of the objective values, different points from different areas of the search space could be selected.

Table X. HV (best, median, std) values obtained for MOEA/D with uniform and proposed initialization methods on ZDT1, ZDT2 and ZDT3 problems

Prob.	Method	best	mean	median	worst	std
ZDT1	Uniform	0.7531	0.7385	0.7410	0.6890	0.0126
	Proposed	**0.7675**	**0.7669**	**0.7669**	**0.7658**	**0.0005**
ZDT2	Uniform	0.3309	0.2790	0.3185	0.0137	0.0940
	Proposed	**0.3416**	**0.3411**	**0.3412**	**0.3399**	**0.0005**
ZDT3	Uniform	0.9824	0.9554	0.9607	0.9023	0.0216
	Proposed	**0.9958**	**0.9886**	**0.9953**	**0.9160**	**0.0217**

The proposed method was incorporated with a well-known DE algorithm and tested on a set of 13 test problems. It was found that, for many problems, optimal or near-optimal solutions were found in the initial population. In addition, the proposed approach was found to be superior to four other initialization mechanisms in terms of the quality of solutions it obtained and its convergence rate which was 43% faster. Also, the initial population mechanism was incorporated with other DE algorithms and used to solve 30 unconstrained problems, with the results indicating the benefits of the proposed method compared with three other techniques. Another finding discussed in this paper was that no single existing initialization mechanism was the best for all DE algorithms, i.e., one might work well with one algorithm but not another. A reason for this might have been the types of search operators and control parameters used in the optimization algorithm.

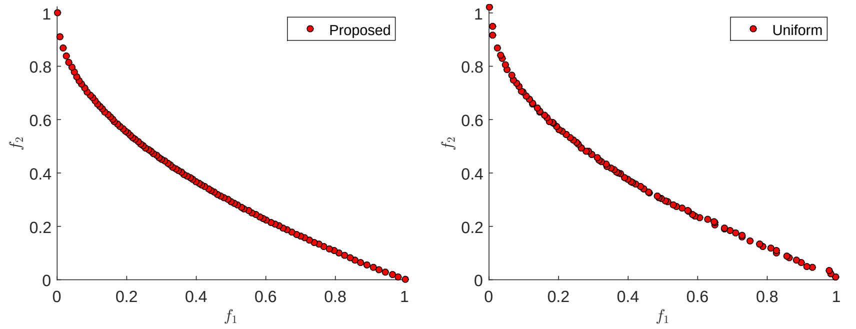
Although the proposed method was the best choice for the algorithms considered in this paper, for some test problems (multi-modal and hybrid functions) it was not, probably because the diversity within the initial population was sufficient. Further investigations into this aspect will be considered as part of our future research. Also, solving real-world black box problems will be another way of analyzing the effectiveness of the proposed technique. Although the selection of points from those generated by the proposed method depends on the problem considered, it will be beneficial to develop a general procedure. Finally, the proposed method's performance for solving expensive optimization problems will be tested.

Acknowledgment

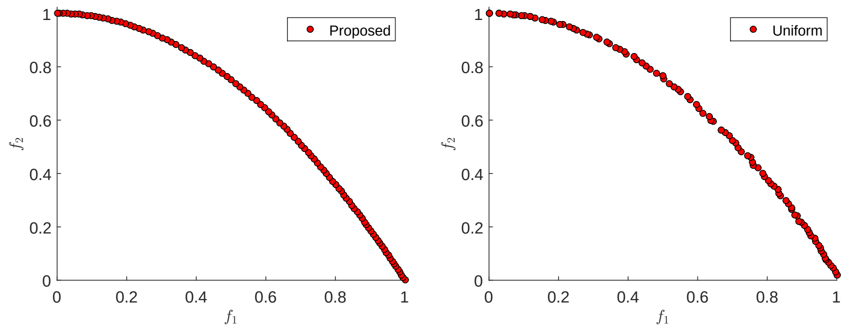
The third author gratefully acknowledges support from CONACyT project no. 221551.

References

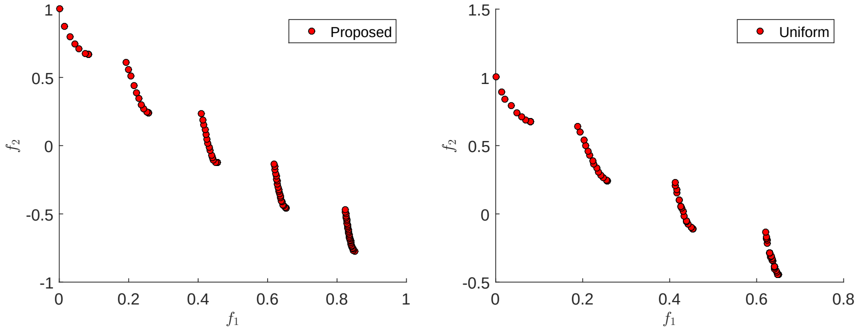
[1] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization.* Springer Science & Business Media, 2006.



(a) ZDT1



(b) ZDT2



(c) ZDT3

Figure 4. Plot of the nondominated front with the maximum HV value found by MOEA/D-proposed (left) and MOEA/D-Uniform (right) for ZDT1, ZDT2 and ZDT3

[2] S. M. Elsayed, R. A. Sarker, and D. L. Essam, “Multi-operator based evolutionary algorithms for solving constrained optimization problems,” *Computers & operations research*, vol. 38, no. 12, pp. 1877–1896, 2011.

[3] D. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.

[4] L. Davis *et al.*, *Handbook of genetic algorithms*. Van Nostrand Reinhold New York, 1991, vol. 115.

[5] R. Storn and K. Price, “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[6] N. Hansen, S. Müller, and P. Koumoutsakos, “Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es),” *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[7] J. Kennedy, “Particle swarm optimization,” in *Encyclopedia of Machine Learning*. Springer, 2010, pp. 760–766.

[8] R. Sarker, J. Kamruzzaman, and C. Newton, “Evolutionary optimization (evopt): a brief review and analysis,” *International Journal of Computational Intelligence and Applications*, vol. 3, no. 04, pp. 311–330, 2003.

[9] Y.-W. Leung and Y. Wang, “An orthogonal genetic algorithm with quantization for global numerical optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 41–53, 2001.

[10] R. Sarker, S. Elsayed, and T. Ray, “Differential evolution with dynamic parameters selection for optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 5, pp. 689–707, Oct 2014.

[11] G. Karafotias, M. Hoogendoorn, and A. Eiben, “Parameter control in evolutionary algorithms: Trends and challenges,” *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 167–187, April 2015.

[12] S. M. Elsayed, R. A. Sarker, and D. L. Essam, “Adaptive configuration of evolutionary algorithms for constrained optimization,” *Applied Mathematics and Computation*, vol. 222, pp. 680–711, 2013.

[13] Q. Fan and X. Yan, “Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies,” *IEEE Transactions on Cybernetics*, vol. 46, no. 1, pp. 219–232, Jan 2016.

[14] M. Yang, C. Li, Z. Cai, and J. Guan, “Differential evolution with auto-enhanced population diversity,” *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 302–315, Feb 2015.

[15] M. Z. Ali, N. H. Awad, P. N. Suganthan, and R. G. Reynolds, “An adaptive multipopulation differential evolution with dynamic population reduction,” *IEEE Transactions on Cybernetics*, vol. PP, no. 99, pp. 1–12, 2016.

[16] X. Qiu, K. C. Tan, and J. X. Xu, "Multiple exponential recombination for differential evolution," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, pp. 1–12, 2016.

[17] S. Kimura and K. Matsumura, "Genetic algorithms using low-discrepancy sequences," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, 2005, pp. 1341–1346.

[18] Z. Ma and G. A. Vandenbosch, "Impact of random number generators on the performance of particle swarm optimization in antenna design," in *6th European Conference on Antennas and Propagation (EUCAP)*. IEEE, 2012, pp. 925–929.

[19] B. Kazimipour, X. Li, and A. Qin, "A review of population initialization techniques for evolutionary algorithms," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, July 2014, pp. 2585–2592.

[20] A. B. Ozer, "Cide: chaotically initialized differential evolution," *Expert Systems with Applications*, vol. 37, no. 6, pp. 4632–4641, 2010.

[21] S. Rahnamayan, H. R. Tizhoosh, and M. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.

[22] E. K. Burke, J. P. Newall, and R. F. Weare, "Initialization strategies and diversity in evolutionary timetabling," *Evolutionary computation*, vol. 6, no. 1, pp. 81–103, 1998.

[23] S. K. Park and K. W. Miller, "Random number generators: good ones are hard to find," *Communications of the ACM*, vol. 31, no. 10, pp. 1192–1201, 1988.

[24] H. Maaranen, K. Miettinen, and M. M. Mäkelä, "Quasi-random initial population for genetic algorithms," *Computers & Mathematics with Applications*, vol. 47, no. 12, pp. 1885–1895, 2004.

[25] S. Helwig and R. Wanka, "Theoretical analysis of initial particle swarm behavior," in *Parallel Problem Solving from Nature*. Springer, 2008, pp. 889–898.

[26] H. G. Schuster and W. Just, *Deterministic chaos: an introduction*. John Wiley & Sons, 2006.

[27] Y. Wang and K. T. Fang, "A note on uniform distribution and experimental design," *KeXue TongBao*, vol. 26, no. 485, p. e9, 1981.

[28] W. Gong, Z. Cai, and L. Jiang, "Enhancing the performance of differential evolution using orthogonal design method," *Applied Mathematics and Computation*, vol. 206, no. 1, pp. 56–69, 2008.

[29] Z.-H. Zhan, J. Zhang, Y. Li, and Y. hui Shi, "Orthogonal learning particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 832–847, Dec 2011.

[30] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Statistical science*, pp. 409–423, 1989.

[31] M. Zaman, S. Elsayed, T. Ray, and R. Sarker, "Evolutionary algorithms for dynamic economic dispatch problems," *IEEE Transactions on Power Systems*, vol. PP, no. 99, pp. 1–10, 2015.

[32] S. M. Elsayed, T. Ray, R. Sarker *et al.*, "A surrogate-assisted differential evolution algorithm with dynamic parameters selection for solving expensive optimization problems," in *IEEE Congress on Evolutionary Computation*. IEEE, 2014, pp. 1062–1068.

[33] S. Rahnamayan and G. G. Wang, "Center-based sampling for population-based algorithms," in *IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 933–938.

[34] H. Wang, Z. Wu, J. Wang, X. Dong, S. Yu, and C. Chen, "A new population initialization method based on space transformation search," in *Fifth International Conference on Natural Computation*, vol. 5. IEEE, 2009, pp. 332–336.

[35] Q. Xu, N. Wang, and R. Fei, "Influence of dimensionality and population size on opposition-based differential evolution using the current optimum," *Information Technology Journal*, vol. 12, no. 1, p. 105, 2013.

[36] S. Y. Park and J. J. Lee, "Stochastic opposition-based learning using a beta distribution in differential evolution," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2184–2194, Oct 2016.

[37] R. Storn and K. Price, *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*. ICSI Berkeley, 1995, vol. 3.

[38] R. Storn, "Differential evolution research-trends and open questions," in *Advances in Differential Evolution*, ser. Studies in Computational Intelligence, U. Chakraborty, Ed. Springer Berlin Heidelberg, 2008, vol. 143, pp. 1–31.

[39] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[40] J. Zhang and A. C. Sanderson, "Jade: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[41] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[42] D. Zaharie, "A comparative analysis of crossover variants in differential evolution," *Proceedings of IMCSIT*, pp. 171–181, 2007.

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

[44] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

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

[46] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.

[47] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *IEEE Congress on Evolutionary Computation (CEC)*, 2013, pp. 71–78.

[48] R. Tanabe and A. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *IEEE Congress on Evolutionary Computation (CEC)*, July 2014, pp. 1658–1665.

[49] G. W. Corder and D. I. Foreman, *Nonparametric statistics for non-statisticians: a step-by-step approach*. John Wiley & Sons, 2009.

[50] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "A new genetic algorithm for solving optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 27, pp. 57–69, 2014.

[51] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[52] S. M. Elsayed, R. A. Sarker, D. L. Essam, and N. M. Hamza, "Testing united multi-operator evolutionary algorithms on the cec2014 real-parameter numerical optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2014, pp. 1650–1657.

[53] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.

[54] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[55] E. Zitzler, D. Brockhoff, and L. Thiele, "The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2007, pp. 862–876.

APPENDIX

All appendices can be found in the supplementary materials attached with the paper, or the personal websites of the authors.

Saber Elsayed (M'10) received the Ph.D. degree in Computer Science from the University of New South Wales Canberra, Australia, in 2012. Currently, Saber is a Research Associate with the School of Engineering and Information Technology, University of New South Wales Canberra. His research interests include the areas of evolutionary algorithms, constraint-handling techniques for evolutionary algorithms, scheduling, big data and cybersecurity using computational intelligence. Dr. Elsayed was the winner of several IEEE-CEC competitions. He is an editorial board member of the International Journal of Business Intelligence and Data Mining and serves as a reviewer in several international journals. Saber was a member of the program committee of several international conferences.

Ruhul Sarker (M'03) received his Ph.D. degree from Dalhousie University, Halifax, Canada, in 1992. He is currently a Professor in the School of Engineering and Information Technology and the Director of Faculty Postgraduate Research, University of New South Wales, Canberra Campus, Australia. His main research interests are evolutionary optimization and applied operations research. He is the lead author of the book Optimization Modelling: A Practical Approach (CRC, Boca Raton, FL, 2007). He has published more than 250 refereed articles in the international journals, edited books, and conference proceedings. Prof. Sarker is currently an Associate Editor of the Memetic Computing Journal, Journal of Industrial and Management Optimization (JIMO), and Flexible Service and Manufacturing Journal (FSMJ)..

Carlos A. Coello Coello (M'98–SM'04–F'11) received the Ph.D. degree in Computer Science from Tulane University, New Orleans, Louisiana, USA, in 1996. He is a Professor (CINVESTAV-3F Researcher) with the Department of Computer Science, CINVESTAV-IPN, Mexico City, Mexico. He has authored or co-authored over 450 technical papers and book chapters. He has also co-authored a book entitled Evolutionary Algorithms for Solving Multi-Objective Problems (Springer, 2007). His publications currently report over 33 000 Google Scholar citations and an H-index of 70. His research interests include evolutionary multiobjective optimization and constraint-handling techniques for evolutionary algorithms. Dr. Coello Coello received the 2007 National Research Award from the Mexican Academy of Sciences in the area of Exact Sciences, the 2013 IEEE Kiyo Tomiyasu Award, and the 2012 National Medal of Science and Arts in the area of Physical, Mathematical, and Natural Sciences. He is an Associate Editor of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and serves on the editorial board of 12 other international journals. He is a member of Association for Computing Machinery and the Mexican Academy of Science.