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

Effects of Population Initialization on Differential Evolution for Large Scale Optimization

Borhan Kazimipour*, Xiaodong Li*, A. K. Qin*†
*School of Computer Science and Information Technology, RMIT University, Melbourne, 3000, Victoria, Australia
Email:{borhan.kazimipour, xiaodong.li, kai.qin}@rmit.edu.au
†School of Automation, Southeast University, Nanjing, China, 210096

***Abstract*—This work provides an in-depth investigation of the effects of population initialization on Differential Evolution (DE) for dealing with large scale optimization problems. Firstly, we conduct a statistical parameter sensitive analysis to study the effects of DE's control parameters on its performance of solving large scale problems. This study reveals the optimal parameter configurations which can lead to the statistically superior performance over the CEC-2013 large-scale test problems. Thus identified optimal parameter configurations interestingly favour much larger population sizes while agreeing with the other parameter settings compared to the most commonly employed parameter configuration. Based on one of the identified optimal configurations and the most commonly used configuration, which only differ in the population size, we investigate the influence of various population initialization techniques on DE's performance. This study indicates that initialization plays a more crucial role in DE with a smaller population size. However, this observation might be the result of insufficient convergence due to the use of a large population size under the limited computational budget, which deserve more investigations.**

I. INTRODUCTION

Differential Evolution (DE) which was originally proposed by Storn and Price, is one of the most effective and efficient stochastic optimization techniques [1]. After about two decades, DE has been developed into one of the most powerful and promising research topics in the field of evolutionary computation [2]. So far, a great and still growing body of literature is devoted to improving its performance [3], explaining its behaviour [4] and expanding its applications in numerous fields [2]. The family of DE variants has presented an exceptional performance when solving challenging optimization problems in different forms [5]. DE related techniques have always been among the best performers in past optimization competitions such as those held at IEEE Congress on Evolutionary Computation (CEC) on single objective, multi-objective and large scale global optimization [6].

Beside its efficiency and effectiveness, DE performance can be significantly affected by its three main control parameters: population size, crossover rate and mutation scale factor. In other words, employing DE with improper parameter configuration will dramatically degrade its efficiency and effectiveness. To lessen this issue, many studies have been done on DE parameters tuning and adaptation [5], [6], [7].

In addition to parameter values, the quality of DE initial population is also reported to have significant impact on its performance. Hence, several advanced population initialization techniques have been proposed and applied to DE [8], [9], [10], [11], [12].

Although the previous studies on DE parameter calibration and population initialization are scientifically valuable, they suffer from some problems. Firstly, to the best of our knowledge, all works on DE parameter calibration are only done on low dimensional problems. In fact, the best parameter configuration for DE in dealing with large scale problems still remain to be discovered. So far, researchers and practitioners use the same values for high dimensional problems as they use for low dimensional ones. To date, there are still no evidences to confirm or decline that the dimensionality of problems has any effect on the best values for DE parameters.

Secondly, all experiments on DE initial population are done using arbitrary parameter values. Indeed, little attention has been taken to study the effects of population initializers while the best parameter configuration is used. Obviously, when improper parameter values are used, the performance of population initialization techniques may be affected and the resulting conclusions may not be precise and practical. This issue, which exists in studies on both low and high dimensional problems, is the source of some contradictions and confusions on the effectiveness of advanced initializers [13], [14].

To fill these gaps in literature, this study conducts a systematic framework to firstly investigate the best parameter configuration of DE when dealing with large scale optimization problems. In fact, this paper investigates whether the best parameter values for low and high dimensions are similar or not.

Secondly, this study compares the effects of some well-known population initialization techniques using the most commonly used and the best found parameter configuration. In other words, this paper investigates whether the quality of initial population has significant influence on DE performance when proper parameter values are used.

Answers to the above mentioned questions will shed more light on the effects of DE parameters and initial population on its performance, especially when handling high dimensional problems. Particularly, researchers and practitioners can use the provided advices for DE parameter calibration in dealing with black-box optimization problems. Moreover, this study clarifies the relation between population initialization and DE main control parameters.

The rest of the paper is organized as follows. The next section presents a brief review of a classic DE algorithm, its main control parameters and some advanced population initialization techniques. Experiments and their results are provided in Section III. Some ideas for future works are given in Section IV. Finally, Section V concludes the paper.

II. Background

In this part a brief review of DE, its main control parameters and some advanced population initialization techniques is provided.

A. Differential Evolution

DE is known as one of the most effective optimizers in dealing with continuous problems. Numerous variants of DE have been proposed and applied on a variety of real-world applications. DE, like other Evolutionary Algorithms (EAs), is designed to deal with black-box optimization problems [15].

Without loss of generality, an optimization (minimization, here) problem can be defined as:

$$\mathbf{x}^* = arg \min_{\mathrm{x} \in \mathcal{R}^{\mathcal{D}}} f(x), \quad f(x) \in \mathcal{R} \tag{1}$$

where $x = \{x^1, x^2, \ldots, x^D\}$ is a D-dimensional vector of decision variables and $f(x) \in \mathcal{R}$ is a real-valued objective function. Note that the exact formula of $f(x)$ in not available in black-box problems. In EA literature, problems with large number of decision variables (e.g. $D > 100$) are widely referred to as large scale problems.

Similar to the majority of EAs, DE is also a *population-based* algorithm. This means, DE is initially starts with a set of decision variables (i.e. initial population) which are usually drawn randomly from the uniform distribution within the solution space (see Section II-B for more details). Then, DE operators are applied to each individual in the population (as parents) to produce another population (as offspring). Both populations then evaluated using the objective function (see Equation 1) and each parent is substituted by its offspring if the offspring is fitter (in terms of objective value) than its parent. The reproduction, evaluation and selection steps are repeated iteratively till termination criteria have met (usually maximum number of objective function evaluations). Finally, the fittest individual of the population from the last iteration is returned as the solution of the optimization problem.

In each iteration of DE process, three operators are applied to each single individual (so-called *target* vector): *mutation*, *recombination* and *selection*. These operators work as follows:

1) Mutation: For each target vector, a *base* vector is chosen from the current population members and added to the scaled difference of a few randomly selected members. The resulting vector is usually called *mutant* vector.

$$\vec{y}_i^t = \vec{x}_b^t + F \times (\vec{x}_{r_1}^t - \vec{x}_{r_2}^t),\, i = 1, 2, \ldots, \text{NP} \tag{2}$$

where r_1 and r_2 are randomly chosen from [1,NP] and t denotes the iteration number. $\vec{x}_b$ and $\vec{y}_i$ are base and mutant vectors, respectively. Note that base and mutant vectors may be produced differently according to various mutation strategies.

2) Recombination: After mutation (see Equation 2), a *trial* vector is generated by the combination of mutant and target vectors.

$$\vec{z}_{i,j}^t = \begin{cases} \vec{y}_{i,j}^t, & \text{if}(rand(0,1) \leq \text{CR} \mid j = j_r) \\ \vec{x^t}_{i,j}, & \text{otherwise} \end{cases} \tag{3}$$

where $\vec{z}_i$ is the trial vector of i^{th} target vector, j indicates the j^{th} variable of vectors, j_r is chosen randomly from [1,NP] and $rand(0,1)$ generate a random number from [0,1]. Note that among several DE recombination schemes, discrete recombinations (a.k.a. crossovers) are the most widely used.

3) Selection: Finally, each target vector (as the main parent and the corresponding trial (as the its offspring) are compared based on their fitness values. The fittest one (e.g. with the smallest objective value in a minimization problem) is selected to remain in the population.

$$\vec{x}_i^{t+1} = \begin{cases} \vec{z}_i^t, & \text{if} f(\vec{x}_i^t) > f(\vec{z}_i^t) \\ \vec{x^t}_i, & \text{otherwise} \end{cases} \tag{4}$$

These three DE operators are iteratively applied to all population members in a round-robin fashion till termination criteria are met.

DE has many variants which are usually denoted using DE/x/y/z style, where:

- *'x'* defines the base vector generation scheme. For example, 'best' indicates the current fittest member is selected as the base vector, while 'rand' means the base vector is selected randomly.
- *'y'* defines the number of pairs of members used in construction of the difference vector(s) (see Equation 2).
- *'z'* defines the scheme of recombination. For example, 'bin' and 'exp' indicate binomial (uniform) and exponential (circular two-point) crossovers, respectively.

DE in a very general form has three main control parameters:

1) Population size (NP): Like other population-based algorithm, NP plays a crucial role in the efficiency and effectiveness of DE. Large population size potentially increases the population diversity and helps DE to sample more regions, simultaneously. However, when computational budget is limited (which in practice usually is), increasing the population size will decrease the number of iterations (i.e generations) and may result in early termination. In other words, DE may be terminated before the population converges to a desirable point.

2) Crossover rate (CR): In discrete recombination, CR value determines the number of decision variables of each target vector which must be interchanged with the corresponding variables of mutant vector. As a rule of thumb, small CR values can boost convergence speed when a few decision variables are interacting with each others. In turn, large CR values are more effective when lots of decision variables are interacting.

3) Mutation scale factor (F): In DE, the exploration-exploitation balance is controlled by F value. As a rule of thumb, too small F values increase the risk of premature convergence (i.e. converge to an undesirable point), while too large F values decrease the convergence speed that degrades DE efficiency and may result in early termination.

Note that advanced variants of DE may have extra control parameters [16]. So far, several strategies such as fixed [17], control [18] and adaptive [19], [20] schemes are proposed for DE parameter calibration. This work follows the framework suggested in [6] and hence, falls into the fixed scheme group.

B. Population Initialization Techniques

Producing initial population is the first step of all population based algorithms including DE. In this starting step, first estimations of the solution(s) are made. Obviously, staring from a good set of estimates improves optimizer performance and save lots of computational resources [21], [22]. However, in dealing with black-box optimization problems, it is almost impossible to identify the best set of estimates (as initial population) before evaluating the whole optimization process. Consequently, for a long time, researchers mostly use uniformly distributed random numbers as the initial population for DE.

Recently, a large and growing body of literature are devoted to advanced EA population initialization techniques [12], [14]. Indeed, several studies claimed that it is possible to improve EAs (including DE) performance only by employing more advanced initialization techniques [22], [23]. Regarding this finding, several works have been done and reported on population initialization for DE [10], [12].

Generally speaking, population initialization techniques are of different forms and hence varying in many characteristics. Among these techniques, *pseudo-random number generators* (PRNGs) are the most widely used techniques [24], [25]. In fact, most of the experiments on large scale optimization have been done using PRNGs because they are available in every programming language and there is no restriction on the number of points (i.e., NP) or dimension size (i.e., D). Despite the simplicity and popularity of PRNGs, it is claimed that they are not the best available options for population initialization of DE when dealing with large scale problems [12].

Recently, a new family of number generators, widely known as *chaotic number generators* (CNGs), are used to improve the randomness and uniformity of the initial population [26]. Generally, chaotic systems (which CNGs try to reflect their behaviour) are dynamic systems which are very sensitive to their initial conditions. Ergodicity, randomness and unpredictability are the main characteristics of chaotic systems [27]. To produce a chaotic sequence, a proper map is required. For example, the following equation is known as tent map [28]:

$$x_{i,j}^{k+1} = \begin{cases} \mu x_{i,j}^{k} & \text{for } x_{i,j}^{k} < \frac{1}{2} \\ \mu(1 - x_{i,j}^{k}) & \text{for } x_{i,j}^{k} \geq \frac{1}{2} \end{cases} \tag{5}$$

while $x_{i,j}^{k}$ is the j^{th} variable of the i^{th} population member in k^{th} iteration and μ is a positive real constant. If $\mu = 2$ and $x_{i,j}^{0} \in (0, 1)$, tent map will produce chaotic sequences.

Another big family of number generators are those which do not care about randomness of the population, but its uniformity. These algorithms, which are usually deterministic (i.e., produce exactly the same population every time they are evaluated), try to generate points that are evenly distributed over the search space. In many cases (e.g. Sobol set [29] and good lattice point [30]) some theoretical upper-bounds exist on the non-uniformity (a.k.a. discrepancy) of the resulting population. These upper-bounds estimate the expected non-uniformity of population in the worst case scenario.

Several studies claimed that these algorithms can significantly improve EAs performance [25], [31]. The application of deterministic uniform number generators on large scale optimization problems, however, is under doubt. While it is widely believed that uniformity of the initial population in dealing with higher dimensional problems is more critical (because of the sparsity of the population), the performance of these number generators may degrade to undesirable levels when dimensionality grows (a.k.a. 'curse of dimensionality') [13], [32], [33].

Not all population initialization techniques consider randomness or uniformity of the population. A popular group of initializers, which recently attracts extensive attentions, exploits objective function to investigate which estimates have more potentials to improve EA performance in succeeding iterations. *Opposition-based learning* (OBL) technique for population initialization, which originally introduced to DE is the most well-known technique in this group [34]. OBL and its variants, such as *quasi-opposition-based learning* (QOBL) [21], firstly produce a population of size NP. Then using some simple rules, they produce another population so-called opposite population. Finally, the best subset (according to their objective values) from the union of both populations is selected as DE initial population. Several studies confirm that the family of OBL techniques improves DE performance on low and high dimensional problems [35], [16].

EA population initialization techniques are not limited to these groups and examples. Indeed, numerous techniques have been proposed and applied to different problems. Reviewing all published works on this topic demands a comprehensive survey paper which is out of the scope of this study [14].

III. Experiments

This study investigates the effects of population initialization on DE in dealing with large scale problems. For a deep investigation and also finding the interaction of population initialization and DE main control parameters, the experiments are divided into two parts. The first part tries to find the best parameter configuration of a well-known DE model when applied to large scale optimization benchmark functions. The second experiment aims to compare the effects of different population initialization techniques on the same DE model when the most common (according to literature) and the best (based on the findings of the first part) parameter configurations are used.

In all parts of the experiments, DE/rand/1/bin is used because of its simplicity and popularity. This classical DE has been used in similar studies on DE parameter calibration in dealing with low dimensional problems [6]. All experiments are done on the most recently benchmark suite for large scale global optimization (LSGO) which were introduced in CEC-2013 (see Section III-A). Succeeding parts provide more information regarding the benchmarks, experimental setup, obtained results and statistical analysis.

A. Benchmark Functions

The CEC 2013 LSGO benchmark suite is currently the latest proposed benchmark in the field of large scale optimization [36]. The suite consists of 15 continuous functions which are grouped into five distinct categories: fully separable functions ($f_1 - f_3$), partially separable functions with a separable subcomponent ($f_4 - f_7$), partially separable functions with no separable subcomponents ($f_8 - f_{11}$), overlapping functions ($f_{12} - f_{14}$) and one fully non-separable function (f_{15}). All functions have 1000 decision variables, except f_{12} and f_{14} which have 905 variables due to overlapping subcomponents.

To improve previously proposed benchmark suite (e.g. CEC 2008 LSGO [37]), new functions with non-uniform subcomponent sizes and overlapping subcomponents have been added to the recent suite. Moreover, new transformations such as ill-conditioning, symmetry breaking and irregularities have been added to CEC 2013 LSGO suite. More details regarding this suite are available in [36].

B. Experimental Setup

As mentioned earlier, this experiment consists of two parts: DE parameter calibration and population initializations effects. The following paragraphs discuss the experimental setups of all parts in details.

1) Parameter Calibration: In the first part of the experiment, the performance of DE/rand/1/bin on all 15 functions of CEC 2013 LSGO benchmark suite are evaluated using 84 different parameter configurations. These parameter configurations consist of all possible combinations of 14 population sizes (i.e. NP$\in [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250, 300]$), three crossover rates (i.e. CR$\in$ $[0.1, 0.5, 0.9]$) and two mutation scale values (i.e. F$\in$ $[0.5, 0.8]$). These configurations cover most of the advised values in previously published studies [6].

For each function, DE/rand/1/bin with all 84 configurations is run for 51 times. Following the framework used in [6], the ith runs of all configurations initialized by the same seed while ith and jth ($i \neq j$) runs of any configuration differ in initial seed. To be consistent with CEC 2013 LSGO framework, the maximum number of function evaluations is restricted to 3e+06 for all functions.

2) Population Initialization: In the second part of the experiment, the effects of six population initialization techniques on the performance of DE/rand/1/bin on all 15 functions of CEC 2013 LSGO benchmark suite are evaluated. Beside the common random population initializer (i.e., PRNG), five well-known and potentially effective initializers are also employed.

Among the selected population initialization techniques, Mersenne Twister [38] and tent map [28] (chosen from PRNG and CNG group, respectively) are stochastic population generators which each run of them (using different initial seed) results in a different population. In contrast, Sobol set (SBL) [29] and good lattice points [30] are selected form the deterministic population generators. They are claimed to be able to provide evenly scattered points with a high level of uniformity. Contrary to those four techniques, OBL [8] and QOBL [21] initializers are chosen from the greedy algorithms who evaluate a big population to select the best subset as the DE initial population. Common values are set for the control parameters of all initialization techniques as suggested and discussed in [12].

In this part of the experiments, two parameter configurations for DE/rand/1/bin are used: the most common parameter configuration ([NP, CR, F]=$[50, 0.9, 0.5]$) and the best configuration found from the first part of the experiment ([NP, CR, F]=$[150, 0.9, 0.5]$).

For each function, DE/rand/1/bin with six population initialization techniques and two configurations is run for 51 times. Similar to the first part, the ith runs of all six initializers use the same initial seed while the initial seeds for runs ith and jth ($i \neq j$) of any technique are chosen differently. The same as the first part, 3e+06 is set for the maximum number of function evaluations.

C. Results and Discussions

The following parts are dedicated to the analysis and discussions of the obtained experimental results.

1) Parameter Calibration: To find the superior parameter configuration(s) for DE/rand/1/bin on CEC 2013 LSGO benchmark among 84 different configurations, some advanced non-parametric statistical tools are employed. In this study, Iman and Davenport test (a.k.a. Friedman rank test) is used to rank the configurations [39]. Based on this ranking, [NP, CR, F]=$[150, 0.9, 0.5]$ is the superior configuration for DE/rand/1/bin on these particular benchmark functions. Table I reports the main statistics of the results obtained using this configuration. For comparison, the results obtained from DE/rand/1/bin using the most common configuration ([NP, CR, F]=$[50, 0.9, 0.5]$), which is also previously reported as the superior configuration on low dimensional problems [6], are added to Table I. As apparent from Table I, the performance of DE/rand/1/bin in most cases is significantly improved when the superior configuration is used.

Note that the ranking is calculated in respect of the configurations performance on all 15 functions. Indeed, we did not compare configurations performance on each single function separately. There are two reasons to avoid such statistical analysis. Firstly, according to [39], at least 252 independent runs of each function (per configuration) are needed to compare 84 algorithms. Otherwise the comparison may not be statistically meaningful. Secondly, we aim to provide some general rules of thumb for users of DE/rand/1/bin to tune its parameters in dealing with black-box problems. Hence, the general assumption is that users have no prior knowledge on the degree of separability of the decision variables of their problems. Consequently, even if superior configuration for each single function of the benchmark suite was reported, users may not be able to find the proper one for their particular application.

To investigate which configurations are significantly dominated by the superior configuration, some post-hoc procedures must be applied. In this study, we employed Li post-hoc procedure [40]. According to [39], Li post-hoc procedure is one of the most powerful procedures among the common statistical tools. Generally, post-hoc procedures compare all methods against a control method to investigate whether they perform significantly different.

In this part of the experiment, each parameter configuration is treated as a single algorithm and the superior configuration (founded by Iman and Davenport ranking) is chosen as the control method. Based on the Li adjusted p-values (not reported here due to limited space), the configurations are divided into two groups: those which their performance significantly differ from (i.e. worse than) the control method (here, [NP, CR, F]=$[150, 0.9, 0.5]$), and those which statistically perform similar to the control configuration. Table II demonstrate the results.

As Table II indicates, the best values for CR and F are 0.9 and 0.5, respectively. This values are consistent with the [6]'s findings for low dimensional problems (i.e., $\mathrm{D} \in [10, 30, 50]$) where these values are founded among the best configurations for DE/rand/1/bin. Table II also reveals that the best range of population size for solving high dimensional problems (with regard to the dedicated computational budget) is between 80 to 250. In comparison with [6]'s findings, which 40 to 60 are suggested as the best range for population size, our results show that DE/rand/1/bin (and probably other EAs) needs more population members when dimension size increases. In other words, while the effective CR and F values for both low and high dimension problems are the same, population size is greatly affected by dimension growth.

It should be noted that these findings are based on the dedicated computational budget (i.e. 3e+06 maximum number of function evaluations). Large increment or decrement of this limit may affect the findings. As an intuitive rule of thumb, providing larger computational budget allows EA to use bigger populations, effectively.

Note that direct comparison between the obtained results from this study and the experiment on low dimensional problems in [6] is impossible. Although both studies follow the same framework and use a similar DE version, since the benchmarks and the computational budgets are greatly different, the direct comparison would be meaningless. However, since both studies provide some general rules of thumb, their findings can be compared.

2) Population Initialization: To study the effects of advanced population initialization techniques on large scale problems, six potentially effective initializers with two configurations are compared. The obtained results from DE/rand/1/bin with the superior ([NP, CR, F]=$[150, 0.9, 0.5]$) and the most common ([NP, CR, F]=$[50, 0.9, 0.f]$) configurations using the six population initialization techniques are reported in Tables III and IV, respectively. As shown in these tables, advanced techniques improved the common PRNG initializer in some functions for both configurations. However, to examine whether these improvements are significant, some statistical tools should be employed (see bellow).

In order to identify superior initialization techniques among six examined algorithms, similar statistical tools (i.e. Iman and Davenport with Li post-hoc procedure) are employed. However, due to the large values of the calculated p-values (i.e. greater than 0.05), the obtained ranks are not statistically reliable while the superior configuration is used. In other words, although some improvements are achieved by employing advanced population initialization techniques, the improvements are not significant from the statistical point of view. This means all population initialization techniques perform statistically similar when they are applied to DE/rand/1/bin with well-tuned control parameters. This is a new finding that indicates when proper values for the control parameters are used, population initialization has only a minor effect on the optimizer performance. In fact, increasing population size from 50 to 150 has more significant impact than changing the initializer algorithm.

This new finding is very important because it challenges the common belief on the effect of population initialization techniques in improvement of EAs on large scale problems [12]. Although some previous works (e.g. [13], [32], [33]) raise doubt about the effectiveness of some techniques in high dimensional spaces, common belief was that advanced techniques can improve EAs performance in both low and high dimensional problems.

It seems the contradiction between the new findings and some of the previous claims is due to two shortcomings of some of previous works. Firstly, to the best of our knowledge, none of the previous studies has tried to compare population initialization techniques on the well-tuned optimizers. Obviously neglecting the significant effects of main control parameters (specially population size) on the performance of EAs may lead the study to weak conclusions.

Secondly, the necessity of employing advanced statistical tools (such as those recommended in [39] and used in this study) for validating the analysis and findings is neglected in most of the previous works. Consequently, some statistically minor improvements of using advanced initializers may wrongly considered as significant contributions.

Note that, although this study is well conducted based on a systematic framework and the findings are statistically validated, the authors are well aware of the need of further investigations to generalise the findings from DE/rand/1/bin to other EAs.

IV. Future Works

As mentioned earlier, the findings from this study challenge the general belief on the potential effects of advanced population initialization techniques on DE/rand/1/bin, when applied to large scale problems. For further generalization of the findings to other EAs, we are interested in repeating this study on other popular EA models. Moreover, considering other performance metrics (besides final objective values) can help us to investigate whether advanced initializers are able to significantly improve EAs according to other aspects.

Furthermore, we believe computational budget has significant effects on the performance of EAs when armed with different population initialization techniques. Consequently, we are interested to study the influence of different computational resource limits on the performance of well-known initializers in dealing with large scale optimization problems.

TABLE I. COMPARISON BETWEEN THE OBTAINED RESULTS FROM THE MOST COMMON ([NP, CR, F]=$[50, 0.9, 0.5]$) AND THE SUPERIOR ([NP, CR, F]=$[150, 0.9, 0.5]$) CONFIGURATIONS FOR DE/RAND/1/BIN ON CEC 2013 LSGO BENCHMARKS. C_C AND C_S STAND FOR THE MOST COMMON AND THE SUPERIOR CONFIGURATIONS, RESPECTIVELY. B, M, W, A AND S STAND FOR THE BEST, MEAN, WORST, AVERAGE (MEAN) AND STANDARD DEVIATION OF 51 INDEPENDENT RUNS, RESPECTIVELY.

Conf.	Stat.	Group 1			Group 2				Group 3				Group 4			Group 5
		F1	**F**2	**F**3	**F**4	**F**5	**F**6	**F**7	**F**8	**F**9	**F**10	**F**11	**F**12	**F**13	**F**14	**F**15
$\mathbf{C_C}$	**B**	5.0e+05	1.9e+04	2.0e+01	1.7e+10	1.5e+06	1.1e+06	5.1e+07	8.9e+13	1.4e+08	9.3e+07	3.9e+10	4.5e+08	4.6e+09	9.6e+10	6.1e+07
	M	4.2e+06	2.0e+04	2.1e+01	4.3e+10	2.2e+06	1.1e+06	1.2e+08	3.5e+14	2.1e+08	9.4e+07	1.5e+11	5.1e+09	8.6e+09	2.0e+11	6.8e+09
	W	3.9e+08	2.2e+04	2.1e+01	9.0e+10	3.2e+06	1.1e+06	8.5e+08	7.9e+14	2.8e+08	9.5e+07	6.4e+11	2.0e+10	1.5e+10	4.3e+11	6.7e+11
	A	3.3e+07	2.0e+04	2.1e+01	4.6e+10	2.3e+06	1.1e+06	1.5e+08	3.7e+14	2.1e+08	9.4e+07	1.7e+11	6.0e+09	8.9e+09	2.1e+11	5.2e+10
	S	8.0e+07	9.2e+02	8.9e-02	1.7e+10	4.1e+05	1.1e+03	1.2e+08	1.7e+14	3.0e+07	2.5e+05	1.1e+11	4.9e+09	2.1e+09	8.4e+10	1.2e+11
$\mathbf{C_S}$	**B**	6.4e+05	7.2e+03	1.1e+01	5.5e+09	7.5e+06	1.8e+01	9.3e+07	2.7e+12	5.4e+07	6.7e+01	5.5e+09	3.0e+07	3.6e+09	4.2e+10	2.8e+07
	M	1.7e+06	8.4e+03	1.3e+01	1.5e+10	8.4e+06	2.0e+01	2.1e+08	1.5e+13	1.4e+08	1.5e+02	4.9e+10	8.3e+07	5.3e+09	8.2e+10	5.5e+07
	W	1.0e+07	9.5e+03	1.4e+01	3.0e+10	8.7e+06	3.8e+01	4.5e+08	1.0e+14	6.5e+08	3.3e+02	1.1e+11	3.8e+09	9.4e+09	1.9e+11	1.2e+08
	A	2.0e+06	8.4e+03	1.3e+01	1.5e+10	8.4e+06	2.0e+01	2.3e+08	2.2e+13	2.7e+08	1.6e+02	5.0e+10	1.8e+08	5.5e+09	8.4e+10	6.1e+07
	S	1.5e+06	5.1e+02	7.1e-01	5.5e+09	2.7e+05	2.9e+00	9.0e+07	1.8e+13	2.3e+08	5.4e+01	2.3e+10	5.3e+08	1.3e+09	2.7e+10	2.0e+07

TABLE II. THE BEST PARAMETER CONFIGURATIONS ACCORDING TO IMAN AND DAVENPORT TEST WITH LI POST-HOC PROCEDURE ON ALL 15 FUNCTIONS OF CEC 2013 LSGO BENCHMARK SUITE. AMONG 84 EXAMINED CONFIGURATIONS, THOSE LEADING TO THE STATISTICALLY BETTER PERFORMANCE WITH 0.05 SIGNIFICANCE LEVEL OVER OTHERS ARE MARKED BY ✓. THE OTHERS ARE STATISTICALLY WORSE THAN THE MARKED CONFIGURATIONS.

Parameters		Population Size													
F	**CR**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**	**150**	**200**	**250**	**300**
0.5	0.1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.5	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.9	-	-	-	-	-	-	-	✓	✓	✓	✓	✓	✓	-
0.8	0.1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.5	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0.9	-	-	-	-	-	-	-	-	-	-	-	-	-	-

TABLE III. THE PERFORMANCE OF SIX POPULATION INITIALIZATION TECHNIQUES USING THE SUPERIOR PARAMETER CONFIGURATION ([NP, CR, F]=$[150, 0.9, 0.5]$) FOR DE/RAND/1/BIN. PRNG, CNG, SBL, GLP, OBL AND QOBL STAND FOR PSEUDO-RANDOM NUMBER GENERATOR, CHAOTIC NUMBER GENERATOR, SOBOL SET, GOOD LATTICE POINTS, OPPOSITION-BASED LEARNING AND QUASI-OPPOSITION-BASED LEARNING, RESPECTIVELY. B, M, W, A AND S STAND FOR THE BEST, MEAN, WORST, AVERAGE AND STANDARD DEVIATION OF 51 RUNS, RESPECTIVELY.

Init.	Stat.	Group 1			Group 2				Group 3				Group 4			Group 5
		F1	**F**2	**F**3	**F**4	**F**5	**F**6	**F**7	**F**8	**F**9	**F**10	**F**11	**F**12	**F**13	**F**14	**F**15
PRNG	**B**	6.4e+05	7.2e+03	1.1e+01	5.5e+09	7.5e+06	1.8e+01	9.3e+07	2.7e+12	5.4e+07	6.7e+01	5.5e+09	3.0e+07	3.6e+09	4.2e+10	2.8e+07
	M	1.7e+06	8.4e+03	1.3e+01	1.5e+10	8.4e+06	2.0e+01	2.1e+08	1.5e+13	1.4e+08	1.5e+02	4.9e+10	8.3e+07	5.3e+09	8.2e+10	5.5e+07
	W	1.0e+07	9.5e+03	1.4e+01	3.0e+10	8.7e+06	3.8e+01	4.5e+08	1.0e+14	6.5e+08	3.3e+02	1.1e+11	3.8e+09	9.4e+09	1.9e+11	1.2e+08
	A	2.0e+06	8.4e+03	1.3e+01	1.5e+10	8.4e+06	2.0e+01	2.3e+08	2.2e+13	2.7e+08	1.6e+02	5.0e+10	1.8e+08	5.5e+09	8.4e+10	6.1e+07
	S	1.5e+06	5.1e+02	7.1e-01	5.5e+09	2.7e+05	2.9e+00	9.0e+07	1.8e+13	2.3e+08	5.4e+01	2.3e+10	5.3e+08	1.3e+09	2.7e+10	2.0e+07
CNG	**B**	5.1e+05	7.3e+03	1.1e+01	4.7e+09	7.7e+06	1.8e+01	1.1e+08	3.1e+12	7.3e+07	7.5e+01	1.2e+10	1.6e+07	3.3e+09	4.3e+10	3.4e+07
	M	1.7e+06	8.3e+03	1.3e+01	1.5e+10	8.3e+06	2.0e+01	2.2e+08	2.2e+13	1.4e+08	1.5e+02	4.7e+10	8.6e+07	5.7e+09	8.5e+10	6.3e+07
	W	2.2e+07	9.4e+03	1.4e+01	2.8e+10	8.9e+06	3.2e+01	7.4e+08	8.5e+13	6.7e+08	5.4e+02	1.1e+11	7.1e+08	9.0e+09	1.7e+11	1.5e+08
	A	2.7e+06	8.3e+03	1.3e+01	1.4e+10	8.3e+06	2.1e+01	2.3e+08	2.6e+13	3.2e+08	1.6e+02	5.0e+10	1.3e+08	5.8e+09	8.7e+10	6.7e+07
	S	3.9e+06	4.5e+02	6.3e-01	5.2e+09	2.7e+05	2.8e+00	1.1e+08	2.0e+13	2.4e+08	8.4e+01	2.1e+10	1.3e+08	1.2e+09	2.7e+10	2.3e+07
SBL	**B**	6.6e+05	7.2e+03	1.2e+01	4.8e+09	7.5e+06	1.8e+01	1.2e+08	7.8e+11	7.0e+07	7.6e+01	1.0e+10	2.4e+07	3.0e+09	3.3e+10	4.3e+07
	M	1.9e+06	8.5e+03	1.3e+01	1.4e+10	8.4e+06	2.0e+01	2.3e+08	1.9e+13	1.5e+08	1.4e+02	5.3e+10	9.4e+07	5.6e+09	8.4e+10	7.9e+07
	W	1.4e+07	9.5e+03	1.4e+01	2.7e+10	8.9e+06	2.7e+01	6.1e+08	1.0e+14	6.6e+08	4.0e+02	1.3e+11	3.4e+08	7.9e+09	1.5e+11	1.3e+08
	A	2.6e+06	8.5e+03	1.3e+01	1.5e+10	8.3e+06	2.0e+01	2.6e+08	2.5e+13	3.5e+08	1.6e+02	5.8e+10	1.2e+08	5.6e+09	8.8e+10	7.9e+07
	S	2.3e+06	4.8e+02	5.8e-01	5.8e+09	2.9e+05	1.5e+00	1.1e+08	2.1e+13	2.5e+08	6.0e+01	2.6e+10	8.0e+07	1.1e+09	2.5e+10	1.9e+07
GLP	**B**	5.5e+05	7.6e+03	1.2e+01	6.2e+09	7.4e+06	1.8e+01	1.1e+08	4.0e+12	6.1e+07	7.4e+01	1.0e+10	2.2e+07	3.6e+09	4.3e+10	7.8e+07
	M	1.6e+06	8.6e+03	1.3e+01	1.5e+10	8.3e+06	2.1e+01	2.1e+08	1.4e+13	1.4e+08	1.5e+02	4.7e+10	9.5e+07	5.9e+09	8.1e+10	1.4e+08
	W	2.0e+07	9.4e+03	1.5e+01	2.9e+10	8.8e+06	3.0e+01	7.1e+08	9.4e+13	6.6e+08	3.0e+02	1.6e+11	1.1e+09	7.9e+09	1.5e+11	1.9e+08
	A	2.6e+06	8.6e+03	1.3e+01	1.5e+10	8.3e+06	2.1e+01	2.4e+08	2.1e+13	3.2e+08	1.6e+02	5.4e+10	1.6e+08	5.9e+09	8.4e+10	1.4e+08
	S	3.2e+06	4.4e+02	6.8e-01	5.6e+09	3.0e+05	1.8e+00	1.2e+08	1.7e+13	2.4e+08	5.5e+01	2.9e+10	1.8e+08	1.0e+09	2.4e+10	2.2e+07
OBL	**B**	5.8e+05	7.2e+03	1.1e+01	6.5e+09	7.7e+06	1.7e+01	9.6e+07	1.1e+12	7.2e+07	7.6e+01	1.1e+10	2.5e+07	2.9e+09	3.1e+10	3.7e+07
	M	2.0e+06	8.4e+03	1.3e+01	1.5e+10	8.3e+06	2.0e+01	2.2e+08	1.8e+13	1.4e+08	1.6e+02	4.5e+10	7.5e+07	6.0e+09	7.8e+10	6.5e+07
	W	1.1e+07	9.6e+03	1.4e+01	3.5e+10	8.8e+06	2.2e+01	4.1e+08	9.9e+13	6.6e+08	3.3e+02	1.0e+11	9.5e+08	8.4e+09	1.7e+11	1.7e+08
	A	2.3e+06	8.4e+03	1.3e+01	1.6e+10	8.3e+06	2.0e+01	2.2e+08	2.3e+13	3.3e+08	1.7e+02	5.0e+10	1.4e+08	5.8e+09	8.1e+10	7.1e+07
	S	1.9e+06	4.7e+02	6.7e-01	5.8e+09	2.6e+05	1.3e+00	8.3e+07	2.1e+13	2.4e+08	6.5e+01	2.2e+10	1.8e+08	1.2e+09	2.6e+10	2.4e+07
QOBL	**B**	4.5e+05	8.0e+03	1.1e+01	4.3e+09	7.5e+06	1.7e+01	1.1e+08	2.1e+12	6.3e+07	7.1e+01	1.9e+10	2.1e+07	3.5e+09	3.6e+10	3.6e+07
	M	1.6e+06	8.6e+03	1.3e+01	1.3e+10	8.3e+06	2.0e+01	2.3e+08	1.7e+13	1.5e+08	1.6e+02	5.1e+10	8.1e+07	6.0e+09	8.2e+10	6.5e+07
	W	1.6e+07	9.7e+03	1.5e+01	3.3e+10	8.8e+06	3.5e+01	5.3e+08	5.3e+13	6.7e+08	5.6e+02	1.0e+11	9.2e+08	1.1e+10	1.3e+11	1.9e+08
	A	2.5e+06	8.7e+03	1.3e+01	1.5e+10	8.3e+06	2.0e+01	2.4e+08	2.0e+13	3.3e+08	1.6e+02	5.3e+10	1.2e+08	5.9e+09	8.1e+10	7.5e+07
	S	2.7e+06	4.5e+02	6.4e-01	6.4e+09	2.6e+05	2.5e+00	1.0e+08	1.3e+13	2.5e+08	7.2e+01	2.3e+10	1.3e+08	1.3e+09	2.3e+10	2.9e+07

V. CONCLUSION

In this study, the effects of population initialization on the most commonly used DE variant are investigated when solving large scale problems. To achieve a strong conclusion, the best values for main control parameters of DE/rand/1/bin are found among a wide range of configurations which covers nearly all previously advised values. The obtained results indicate the best CR and F values for solving high dimensional problems are almost the same as the previously found values for low dimensional problems. However, the empirical study suggests to increase population size to some threshold values when dimensionality of the problems grows. Indeed, the most commonly used population size is shown to be inferior for dealing with the most of the large scale benchmark functions.

Besides the provided advices for selecting parameter values of DE/rand/1/bin for solving large scale problems, this paper

TABLE IV. THE PERFORMANCE OF SIX POPULATION INITIALIZATION TECHNIQUES USING THE COMMON PARAMETER CONFIGURATION ([NP, CR, F]=$[50, 0.9, 0.5]$) FOR DE/RAND/1/BIN. PRNG, CNG, SBL, GLP, OBL AND QOBL STAND FOR PSEUDO-RANDOM NUMBER GENERATOR, CHAOTIC NUMBER GENERATOR, SOBOL SET, GOOD LATTICE POINTS, OPPOSITION-BASED LEARNING AND QUASI-OPPOSITION-BASED LEARNING, RESPECTIVELY. B, M, W, A AND S STAND FOR THE BEST, MEAN, WORST, AVERAGE AND STANDARD DEVIATION OF 51 RUNS, RESPECTIVELY.

Init.	**Stat.**	**Group 1**			**Group 2**				**Group 3**				**Group 4**			**Group 5**
		F1	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**	**F11**	**F12**	**F13**	**F14**	**F15**
PRNG	**B**	5.0e+05	1.9e+04	2.0e+01	1.7e+10	1.5e+06	1.1e+06	5.1e+07	8.9e+13	1.4e+08	9.3e+07	3.9e+10	4.5e+08	4.6e+09	9.6e+10	6.1e+07
	M	4.2e+06	2.0e+04	2.1e+01	4.3e+10	2.2e+06	1.1e+06	1.2e+08	3.5e+14	2.1e+08	9.4e+07	1.5e+11	5.1e+09	8.6e+09	2.0e+11	6.8e+09
	W	3.9e+08	2.2e+04	2.1e+01	9.0e+10	3.2e+06	1.1e+06	8.5e+08	7.9e+14	2.8e+08	9.5e+07	6.4e+11	2.0e+10	1.5e+10	4.3e+11	6.7e+11
	A	3.3e+07	2.0e+04	2.1e+01	4.6e+10	2.3e+06	1.1e+06	1.5e+08	3.7e+14	2.1e+08	9.4e+07	1.7e+11	6.0e+09	8.9e+09	2.1e+11	5.2e+10
	S	8.0e+07	9.2e+02	8.9e-02	1.7e+10	4.1e+05	1.1e+03	1.2e+08	1.7e+14	3.0e+07	2.5e+05	1.1e+11	4.9e+09	2.1e+09	8.4e+10	1.2e+11
CNG	**B**	7.0e+05	1.7e+04	2.1e+01	2.2e+10	1.5e+06	1.1e+06	5.4e+07	9.8e+13	1.5e+08	9.3e+07	4.2e+10	3.7e+08	4.2e+09	9.7e+10	1.7e+08
	M	5.6e+06	2.0e+04	2.1e+01	4.1e+10	2.2e+06	1.1e+06	1.3e+08	3.4e+14	2.0e+08	9.4e+07	1.5e+11	6.0e+09	8.3e+09	2.0e+11	1.1e+10
	W	2.6e+08	2.3e+04	2.1e+01	7.4e+10	3.1e+06	1.1e+06	4.5e+08	7.8e+14	2.6e+08	9.4e+07	4.6e+11	1.8e+10	1.2e+10	4.9e+11	6.3e+11
	A	2.1e+07	2.0e+04	2.1e+01	4.4e+10	2.2e+06	1.1e+06	1.5e+08	3.4e+14	2.0e+08	9.4e+07	1.8e+11	6.4e+09	8.1e+09	2.2e+11	5.2e+10
	S	4.2e+07	9.7e+02	8.2e-02	1.4e+10	4.0e+05	1.1e+03	8.0e+07	1.7e+14	2.8e+07	2.7e+05	1.1e+11	4.2e+09	1.9e+09	8.6e+10	1.1e+11
SBL	**B**	8.8e+05	1.8e+04	2.0e+01	1.6e+10	1.5e+06	1.1e+06	4.6e+07	1.0e+14	1.4e+08	9.3e+07	6.7e+10	3.9e+08	5.3e+09	8.2e+10	6.6e+06
	M	7.7e+06	2.0e+04	2.1e+01	4.4e+10	2.2e+06	1.1e+06	1.4e+08	3.4e+14	2.2e+08	9.4e+07	1.8e+11	4.2e+09	8.7e+09	2.1e+11	1.0e+07
	W	2.2e+08	2.3e+04	2.1e+01	9.4e+10	3.3e+06	1.1e+06	4.5e+08	6.0e+14	2.8e+08	9.5e+07	4.3e+11	1.1e+10	1.5e+10	4.4e+11	1.5e+07
	A	2.1e+07	2.0e+04	2.1e+01	4.6e+10	2.3e+06	1.1e+06	1.8e+08	3.3e+14	2.1e+08	9.4e+07	2.0e+11	4.8e+09	9.2e+09	2.2e+11	1.0e+07
	S	3.7e+07	1.0e+03	9.9e-02	1.9e+10	4.1e+05	9.4e+02	1.2e+08	1.2e+14	3.4e+07	2.4e+05	9.3e+10	3.2e+09	2.3e+09	8.0e+10	2.0e+06
GLP	**B**	1.0e+06	1.8e+04	2.0e+01	1.8e+10	1.2e+06	1.1e+06	6.2e+07	8.5e+13	1.5e+08	9.4e+07	1.5e+10	1.6e+08	4.0e+09	7.7e+10	7.0e+06
	M	1.1e+07	2.0e+04	2.1e+01	4.5e+10	2.3e+06	1.1e+06	1.3e+08	3.4e+14	2.0e+08	9.4e+07	1.3e+11	4.7e+09	7.5e+09	2.2e+11	9.7e+06
	W	2.0e+08	2.2e+04	2.1e+01	1.2e+11	3.4e+06	1.1e+06	5.1e+08	7.6e+14	3.0e+08	9.5e+07	6.7e+11	1.8e+10	1.5e+10	3.7e+11	2.0e+07
	A	3.6e+07	2.0e+04	2.1e+01	4.7e+10	2.3e+06	1.1e+06	1.6e+08	3.4e+14	2.1e+08	9.4e+07	1.6e+11	6.0e+09	7.9e+09	2.2e+11	1.0e+07
	S	5.1e+07	8.0e+02	1.0e-01	2.0e+10	4.8e+05	1.2e+03	8.5e+07	1.4e+14	3.0e+07	2.2e+05	1.0e+11	4.7e+09	2.4e+09	8.2e+10	2.6e+06
OBL	**B**	8.9e+05	1.8e+04	2.0e+01	1.3e+10	1.2e+06	1.1e+06	4.1e+07	1.1e+14	1.5e+08	9.3e+07	3.6e+10	6.0e+08	3.8e+09	9.9e+10	1.5e+08
	M	4.8e+06	2.0e+04	2.1e+01	4.4e+10	2.3e+06	1.1e+06	1.3e+08	3.0e+14	2.1e+08	9.4e+07	1.6e+11	4.2e+09	8.2e+09	1.9e+11	9.8e+09
	W	9.1e+08	2.3e+04	2.1e+01	8.9e+10	3.3e+06	1.1e+06	7.6e+08	7.4e+14	3.1e+08	9.4e+07	4.5e+11	1.5e+10	1.3e+10	5.1e+11	4.4e+11
	A	5.5e+07	2.0e+04	2.1e+01	4.6e+10	2.3e+06	1.1e+06	1.7e+08	3.3e+14	2.1e+08	9.4e+07	1.7e+11	5.2e+09	8.2e+09	2.2e+11	3.6e+10
	S	4.6e+07	8.0e+02	1.0e-01	1.7e+10	4.1e+05	1.0e+03	1.0e+08	1.4e+14	3.5e+07	2.2e+05	1.4e+11	4.1e+09	1.9e+09	9.0e+10	2.5e+11
QOBL	**B**	6.8e+05	1.9e+04	2.0e+01	1.9e+10	1.4e+06	1.1e+06	5.3e+07	6.5e+13	1.7e+08	9.3e+07	3.6e+10	5.0e+08	4.5e+09	1.1e+11	3.9e+08
	M	4.0e+06	2.1e+04	2.1e+01	4.7e+10	2.2e+06	1.1e+06	1.4e+08	3.3e+14	2.2e+08	9.4e+07	1.8e+11	4.4e+09	7.8e+09	1.8e+11	2.1e+10
	W	2.6e+08	2.2e+04	2.1e+01	9.0e+10	3.4e+06	1.1e+06	5.4e+08	6.5e+14	3.0e+08	9.4e+07	7.3e+11	1.8e+10	1.3e+10	5.5e+11	1.7e+12
	A	1.6e+07	2.0e+04	2.1e+01	4.7e+10	2.3e+06	1.1e+06	1.7e+08	3.4e+14	2.2e+08	9.4e+07	2.2e+11	5.4e+09	8.0e+09	2.0e+11	8.9e+10
	S	4.6e+07	8.0e+02	1.0e-01	1.7e+10	4.1e+05	1.0e+03	1.0e+08	1.4e+14	3.5e+07	2.2e+05	1.4e+11	4.1e+09	1.9e+09	9.0e+10	2.5e+11

studied six well-known population initialization techniques. The obtained results from the potentially effective techniques indicate that some improvements may be expected by employing advanced initializers on large scale problems. However, advanced non-parametric statistical tools show that the improvements are not statistically significant when larger population sizes are used. This new finding challenges the common belief of the effect of advanced initialization techniques on EAs in dealing with large scale problems.

New findings from this study suggest future studies on population initialization techniques to be carried out with extra considerations. Firstly, the significant effects of main control parameters, especially population size, should not be neglected. Secondly, advance statistical tests must be employed to validate the findings. Otherwise, minor enhancements may wrongly be reported as significant improvements. To generalize the findings to other EA models, further investigations are yet required to be done on this topic.

ACKNOWLEDGEMENT

This work was supported by NSFC under Grant No. 61005051, SRFDP under Grant No. 20100092120027 and ARC Discovery Grant (DP120102205).

REFERENCES

[1] 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.

[2] H. A. Abbass, "The self-adaptive pareto differential evolution algorithm," in *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, vol. 1. IEEE, 2002, pp. 831–836.

[3] U. K. Chakraborty, *Advances in differential evolution*. Springer, 2008, vol. 143.

[4] D. Zaharie, "Critical values for the control parameters of differential evolution algorithms," in *Proceedings of MENDEL*, 2002, pp. 62–67.

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

[6] A. Qin and X. Li, "Differential evolution on the cec-2013 single-objective continuous optimization testbed," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 1099–1106.

[7] 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," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 6, pp. 646–657, 2006.

[8] S. Rahnamayan, H. R. Tizhoosh, and M. Salama, "A novel population initialization method for accelerating evolutionary algorithms," *Computers & Mathematics with Applications*, vol. 53, no. 10, pp. 1605–1614, 2007.

[9] L. Peng, Y. Wang, and G. Dai, "Ude: differential evolution with uniform design," in *Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on*. IEEE, 2010, pp. 239–246.

[10] M. Ali, M. Pant, and A. Abraham, "Unconventional initialization methods for differential evolution," *Applied Mathematics and Computation*, 2012.

[11] V. V. de Melo and A. C. Botazzo Delbem, "Investigating smart sampling as a population initialization method for differential evolution in continuous problems," *Information Sciences*, vol. 193, pp. 36–53, 2012.

[12] B. Kazimipour, X. Li, and A. Qin, "Initialization methods for large scale global optimization," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 2750–2757.

[13] R. W. Morrison, "Dispersion-based population initialization," in *Genetic and Evolutionary ComputationGECCO 2003*. Springer, 2003, pp. 1210–1221.

[14] B. Kazimipour, X. Li, and A. Qin, "A review of population initialization

techniques for evolutionary algorithms," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014.

[15] B. Kazimipour, B. Salehi, and M. Z. Jahromi, "A novel genetic-based instance selection method: Using a divide and conquer approach," in *Artificial Intelligence and Signal Processing (AISP), 2012 16th CSI International Symposium on*. IEEE, 2012, pp. 397–402.

[16] B. Kazimipour, M. N. Omidvar, X. Li, and A. Qin, "A novel hybridization of opposition-based learning and cooperative co-evolutionary for large-scale optimization," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014.

[17] K. V. Price, R. M. Storn, and J. A. Lampinen, "Differential evolution a practical approach to global optimization," 2005.

[18] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*. ACM, 2005, pp. 991–998.

[19] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 2. IEEE, 2005, pp. 1785–1791.

[20] 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.

[21] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Quasi-oppositional differential evolution," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE, 2007, pp. 2229–2236.

[22] M. Clerc, "Initialisations for particle swarm optimisation," *Online at http://clerc. maurice. free. fr/pso*, 2008.

[23] M. Pant, M. Ali, and V. Singh, "Differential evolution using quadratic interpolation for initializing the population," in *Advance Computing Conference, 2009. IACC 2009. IEEE International*. IEEE, 2009, pp. 375–380.

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

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

[26] A. Gutiérrez, M. Lanza, I. Barriuso, L. Valle, M. Domingo, J. Perez, and J. Basterrechea, "Comparison of different pso initialization techniques for high dimensional search space problems: A test with fss and antenna arrays," in *Antennas and Propagation (EUCAP), Proceedings of the 5th European Conference on*. IEEE, 2011, pp. 965–969.

[27] Y. Gao and Y.-J. Wang, "A memetic differential evolutionary algorithm for high dimensional functions' optimization," in *Natural Computation, 2007. ICNC 2007. Third International Conference on*, vol. 4. IEEE, 2007, pp. 188–192.

[28] N. Dong, C.-H. Wu, W.-H. Ip, Z.-Q. Chen, C.-Y. Chan, and K.-L. Yung, "An opposition-based chaotic ga/pso hybrid algorithm and its application in circle detection," *Computers & Mathematics with Applications*, vol. 64, no. 6, pp. 1886–1902, 2012.

[29] P. Bratley and B. L. Fox, "Algorithm 659: Implementing sobol's quasirandom sequence generator," *ACM Transactions on Mathematical Software (TOMS)*, vol. 14, no. 1, pp. 88–100, 1988.

[30] I. H. Sloan and S. Joe, *Lattice methods for multiple integration*. Oxford University Press, 1994.

[31] 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.

[32] W. J. Morokoff and R. E. Caflisch, "Quasi-random sequences and their discrepancies," *SIAM Journal on Scientific Computing*, vol. 15, no. 6, pp. 1251–1279, 1994.

[33] X. Wang and I. H. Sloan, "Low discrepancy sequences in high dimensions: How well are their projections distributed?" *Journal of Computational and Applied Mathematics*, vol. 213, no. 2, pp. 366–386, 2008.

[34] Q. Xu, L. Wang, N. Wang, X. Hei, and L. Zhao, "A review of opposition-based learning from 2005 to 2012," *Engineering Applications of Artificial Intelligence*, 2014.

[35] S. Rahnamayan, H. R. Tizhoosh, and M. Salama, "Opposition versus randomness in soft computing techniques," *Applied Soft Computing*, vol. 8, no. 2, pp. 906–918, 2008.

[36] X. Li, K. Tang, M. N. Omidvar, Z. Yang, K. Qin, and H. China, "Benchmark functions for the cec2013 special session and competition on large-scale global optimization," *gene*, vol. 7, p. 33, 2013.

[37] K. Tang, X. Yáo, P. N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, and Z. Yang, "Benchmark functions for the cec2008 special session and competition on large scale global optimization," *Nature Inspired Computation and Applications Laboratory, USTC, China*, 2007.

[38] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 8, no. 1, pp. 3–30, 1998.

[39] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[40] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.