2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

Initialization Methods for Large Scale Global Optimization

Borhan Kazimipour*, Xiaodong Li*, A. K. Qin*†
*School of Computer Science and Information Technology, RMIT University, Melbourne, 3001, Victoria, Australia
Email:{borhan.kazimipour, xiaodong.li, kai.qin}@rmit.edu.au
†School of Automation, Southeast University, Nanjing, China, 210096

***Abstract*—Several population initialization methods for evolutionary algorithms (EAs) have been proposed previously. This paper categorizes the most well-known initialization methods and studies the effect of them on large scale global optimization problems. Experimental results indicate that the optimization of large scale problems using EAs is more sensitive to the initial population than optimizing lower dimensional problems. Statistical analysis of results show that basic random number generators, which are the most commonly used method for population initialization in EAs, lead to the inferior performance. Furthermore, our study shows, regardless of the size of the initial population, choosing a proper initialization method is vital for solving large scale problems.**

I. Introduction

Evolutionary algorithms (EAs), like other population-based optimization algorithms, rely on the initial population consisting of potential solutions. Traditionally, basic random number generators (RNGs) are widely used to initialize the population of EAs. For years, researchers did not pay much attention to the potential influence of population initialization on the performance of EAs. Recent studies suggest that it is possible to significantly improve the performance of EAs just by using different initialization methods [1]. Moreover, a large and growing body of literatures has proposed new ways of generating better initial populations [2], [3]. Some of these methods are known to be more random [4], [5], [6], more uniform [3], [7], [8] or to some extent more informed than RNGs [2], [9], [10], [11]. Several published studies have revealed that the more advanced methods can increase the probability of finding the optimum solution, decrease the computational cost [12], reduce the variance of the results [13] and improve the solution quality of EAs [14].

According to the aforementioned literature, advanced initialization methods for EAs have been widely used to solve low and medium dimensional problems. They are, however, not widely used to deal with large scale global optimization (LSGO) problems which usually have more than 100 decision variables. As a fact, all participants of previous competitions on LSGO held at the IEEE Congress on Evolutionary Computation (CEC-2008, CEC-2010 and CEC-2012), except [15], simply used RNGs as initialization methods. Considering the growing demands on solving LSGO problems [16], [17], it is important to investigate whether advanced initialization methods are able to improve the performance of state-of-the-art EAs in comparison to basic RNGs.

Up to now, a few comparative studies have been carried out on the effect of different initialization methods on the optimization performance [1], [12], [14]. Although previously published comparative studies are scientifically informative, they suffer from several problems. Firstly, to our best knowledge, they are limited to investigate just a few methods mostly from the same category. In fact, no one compared more than four initialization methods or studied several different categories of methods for generating initial populations. Another limitation of the previous works is that all of them only studied low dimensional problems. As a matter of fact, less existing works addressed the problems of the dimension size larger than 60. It is also important to mention that previous studies rarely investigated the relation between the population size and the initialization method. Last but not least, a number of works studied the effect of initialization methods on very specific problems. These studies and the corresponding conclusions cannot be generalized well to problems of the very different nature. This paper, however, aims to fill these gaps and provide a more comprehensive comparative study on the influence of population initialization on LSGO problems.

In this paper, we specifically aim to highlight the importance of choosing proper initialization methods when dealing with LSGO problems using the state-of-the-art EAs. We want to demonstrate that there exist some initialization methods, which can perform consistently better than RNGs regardless of the problem dimension size or the size of initial population. In order to conduct a comprehensive comparative study, we categorize existing methods into several groups and select a few methods from each group as representatives to compare. This categorization can help researchers to perform deeper analysis.

In the next section, we review and categorize the literature of initialization methods for EAs. In Section III, two state-of-the-art representatives from each category are introduced in more details. The experimental setup and results are discussed in Section IV and V, respectively. Finally, Section VI concludes the paper.

II. Literature Review

Regardless of the type of the EA method under consideration, at least one stage is always in common: population initialization. Every population-based optimizer demands an initial population at the early step of the algorithm. Besides Random Number Generators (RNGs), as the most common initialization method, many promising alternatives have recently been proposed. So far, however, there has been little agreement on the possible effect of these advanced methods on high dimensional problems. For example, study in [18]

found that uniformity of the initial population plays a more important role in higher dimensional problems (up to 50 dimensions) while [13], in contrast, claim that uniform initialization methods lose their effectiveness in problems with dimensionality larger than 12. The contradiction and confusion such as this and other limitations of previous comparative studies (described in Section I), motivate us to systematically examine the influences of advanced initialization methods on LSGO problems under different population size conditions.

Since the recently proposed methods are of very different forms and vary in characteristics, we group them into five major categories:

1) Stochastic methods
2) Deterministic methods
3) Two-step methods
4) Hybrid methods
5) Application specific methods

Each category may also comprise several minor subcategories. Although other types of grouping were possible, we find above classification more comprehensive and consistent.

Generally speaking, the main aim of methods in the first group (i.e. Stochastic methods) is to produce random numbers [19], [20]. In the designers' point of view, more random sequence (and hence less predictable and reproducible set) makes better initial population for stochastic algorithms like EAs. Accordingly, the most important attribute of stochastic methods is cycle time (i.e. period length) and the degree of unpredictability [21]. Although these methods cannot produce "true random" numbers, they are the most used initialization methods in EAs.

Recently, a subcategory of stochastic methods which is called Chaotic number generators attracts more interest among researchers [20]. Chaotic methods mimic the behaviour of dynamical systems which yields generally unpredictable point sequences [4].

In contrast to the first group, Deterministic methods focus more on the uniformity rather than randomness [13]. Many researchers believe, in absence of prior knowledge about the problem, a more uniform initial population enhances EA's exploration ability in the early iterations [12]. In other words, increasing the uniformity of the initial population reduces the probability of missing a large part of search space and saves lots of computational budget.

In theory, a wide range of methods can be classified in the group of deterministic methods. Some of these methods have unique features such as orthogonality [7] or regularity [22] while the others only differ in the way of generating uniform points [23]. A few types of deterministic methods like quasi sequence generators [24] and low-discrepancy sequences [1], [12], [25] have the support of theoretical upper-bounds on discrepancy (i.e. non-uniformity). Most of these methods need very large prime numbers to work properly in high dimensions. Other types of deterministic uniform number generators are mainly borrowed from Experimental Design concepts [3], [7], [8]. These methods are generally iterative and demand a large amount of memory and long time to compute on very high dimensions.

TABLE I. SURVEY OF PREVIOUS COMPARATIVE STUDIES
S, D, T, H AND A STAND FOR STOCHASTIC, DETERMINISTIC, TWO-STEP, HYBRID AND APPLICATION SPECIFIC, RESPECTIVELY.

Author(s)	Ref.	Dim.	Year	Alg.	S	D	T	H	A
Clerc	[1]	30	2008	PSO	*				
Peng et al	[8]	50	2012	DE	*	*	*		
Wang et al	[10]	30	2009	PSO	*		*		
de Melo et al	[11]	60	2012	DE	*		*		
Kimura et al	[12]	20	2005	GA	*	*			
Richards et al	[18]	50	2004	PSO	*	*			
Maaranen et al	[24]	50	2004	GA	*	*			
Uy et al	[25]	40	2007	PSO	*	*			
Gutierrez et al	[28]	30	2011	PSO	*	*	*		*
Chou et al	[29]	50	2000	GA	*	*			
Khanum et al	[30]	30	2011	DE	*		*		
Pant et al	[31]	30	2009	DE	*		*		
Maaranen et al	[32]	50	2007	GA	*	*			

In recent years, there has been an increasing amount of literature on Two-step initialization methods [2], [9], [11]. These methods basically generate initial points (first phase) and then try to enhance them according to some criteria (second phase). Most of two-step methods exploit fitness function to estimate search space characteristics. These methods tend to generate points in promising regions and do not care about randomness or uniformity of the population. All variations of Opposition Based Learning [2], [9], [10] and Smart Sampling [11] belong to this category. These methods consume a portion of computational budget to guide the algorithm to better regions. The performance of two-step methods also depends on the original population generated in the first phase [26], [27]. In fact, they are almost like greedy searches which are only applied to the first iteration of EAs.

Hybrid methods, in general, are those methods which are combinations of several basic methods [20], [26], [27]. They may inherit the pros and cons of the basic methods which they are made from. We believe studying basic methods can shed more light on hybrid methods as well. If our knowledge about basic methods is insufficient, studying hybrid methods would provide little benefit and interest.

Finally, the application specific group comprises methods which are designed to be specifically applied to a few particular real-world problems [4], [14], [28]. These methods may be very promising in some cases, however, they are not applicable in other areas. Consequently, study on these group of methods must be done by researchers who are expert on those specific domains.

Besides that part of the literature which introduced original initialization methods, a few comparative studies on existing methods have also been published. As mentioned earlier, these studies are largely focussed on low dimensional problems and limited to a few methods mostly selected from the same category. Table I surveys these previously published studies where S, D, T, H and A stand for Stochastic, Deterministic, Two-step, Hybrid and Application specific, respectively.

III. INITIALIZATION METHODS

As mentioned in former sections, the research goal of this study is to investigate relationship between initial population

and final outcome of EAs specifically on LSGO problems. To do so, we categorized existing methods into five major categories (see Section II). In this section, the most common methods from each category are selected and discussed in more details. Nonetheless, extra details and unpopular, discrete, application specific and hybrid methods are excluded to avoid obscuration. This exclusion results to better concentration on the basic real-valued methods which are more general and well-known. Interested readers are highly encouraged to follow the cited references and references therein.

A. Stochastic Methods

Pseudo-Random Number Generators or simply RNGs are the most commonly used methods for initialization of EAs for many years. These methods attempt to generate statistically uniform random numbers within the given range [14]. In reality, however, RNGs cannot produce a perfect uniform distribution of points [24]. This shortcoming gets worse when the dimensionality of the search space grows or the number of points diminishes. Therefore, we expect on LSGO problems RNGs lose their effectiveness because the dimension of the search space is very high and the population size is not large enough to sample all areas. Generally speaking, different algorithms and implementations of RNG methods may affect the quality of the generated points [21], [14]. In this study we simply use the default *rand* function [19] of Octave version 3.6.2.

Beside RNG, chaotic methods are also employed as random population generators [5], [6]. Theoretically, chaotic motion can traverse every state in a certain region by its own regularity. Hence, chaotic initialization methods can form better distributions in the search space due to the randomness and non-repetitive ergodicity of chaos [4].

To produce an initial population which mimics a chaotic system, a proper map is required. Previous studies exploited Tent, Logistic and Sinusoidal maps [4], [5], [6]. Tent map (TNT), for example, works as follows:

$$x_{i,j}^{(k+1)} = \mu(1 - 2|x_{i,j}^{(k)} - 0.5|), \quad 0 \leq x_{i,j}^{(0)} \leq 1. \qquad (1)$$

where $x_{i,j}^{(k)}$ is jth variable of ith individual in kth iteration and μ is the bifurcation factor [4].

As is shown in Equation 1, chaotic methods are deterministic. Nevertheless, the resulting chaotic sequences are highly sensitive to the initial condition and their outputs are not predictable. Here, we follow the implementation of Tent map presented in [4].

B. Deterministic Methods

Generally speaking, deterministic methods aim to produce a uniform population using geometrical techniques. In theory, a wide range of population initialization methods can be classified as deterministic methods. We divide this category into two subcategories: Low Discrepancy (LD) and Experimental Design (ED) methods. The main difference between these two subcategories is that LD methods have the support of theoretical upper-bounds on discrepancy (i.e. non-uniformity) while ED methods are not supported by such mathematical bounds. Technically, upper-bound on non-uniformity shows how good (or bad) a method can perform in extreme cases [24]. In practice, these theoretical limits are not valid on LSGO problems due to the unsatisfied basic assumptions (i.e. The population size must be very large in order for those upper-bounds to be valid.). These two subcategories are discussed below.

Low Discrepancy Methods: As mentioned previously, discrepancy is a measure of non-uniformity. According to theoretical studies, LD points are more uniform than points generated by RNGs [25]. Therefore, LD methods are mainly employed to generate sets of evenly distributed points (rather than randomly distributed).

A very popular type of LD point generator is Quasi Random Sequence (QRS) which has great applications in Quasi-Monte Carlo integrations [24]. Nevertheless these methods are called quasi-random or sub-random sequences, they are deterministic by nature which means no randomness is involved in their algorithms. Hence, QRSs always produce the same sequences from a given dimension and population size. Some researchers, however, have used random start QRSs to add some randomness to the resulting sequences [12]. In EA literature, several QRS sets such as Halton, Sobol, Niederreiter and Faure sets have been widely used. In this study we use Matlab (version 7.11) official implementation of Sobol (SBL) set [23].

Another LD method which is known to produce very regular distribution is Good Lattice Point (GLP). The GLP set is also deterministic and uniform; and has been widely used in Monte Carlo integrations. In this study, we follow the implementation of GLP introduced in [22].

Experimental Design Methods: Experimental design (ED) methods are another types of deterministic uniform point generators. Since ED methods mostly produce discrete numbers, some post-processing must be done before it can be applied to real-value problems. A very common ED method is Uniform Design (UD) which as a space-filling method seeks points uniformly scattered on the domain [8]. According to [3], UD initialization method can accelerate convergence speed and improve the stability of EAs.

Suppose we want to generate a uniform initial population of size M in D dimensional hypercube using UD. Let $P = (p_1, p_2, ...p_D)$ be D prime numbers smaller than M which are selected randomly, then $X_{i,j} = (i \times p_j) \ mod \ M$ is the jth variable of ith initial individual. Note that the number of prime numbers fewer than M should be larger than or equal to D. Otherwise, we have to increase the population size.

Orthogonal Design (OD) is another experimental design method which has been applied to produce evenly scattered points over the search space [7], [8]. Technically, OD produces an orthogonal 2D array like $L_M(Q^N)$ where M is the number of rows (i.e. population size), N is the number of columns (i.e. factors), Q is the levels and L denotes Latin square. According to [7], Q should be an even integer and $M = Q^J$ while J is a positive integer satisfying $N = \frac{Q^J - 1}{Q - 1}$. The main attributes of an orthogonal array are as follows:

1) For the factor in any column, every level occurs exactly $\frac{M}{Q}$ times. This attribute make the resulting population very uniform.

2) Orthogonality of the array is not sensitive to the number or the order of columns. Therefore, one can reorder the columns or remove a number of them and the resulting array is still orthogonal.

This study follows the implementation of OD presented in [7].

C. Two-step Methods

Two-step methods are those algorithms which generate an initial population in the first step and then try to improve them using some guidelines (e.g., fitness function) in the second step. These methods can be seen as greedy methods which are applied only in the first iteration.

Undoubtedly, the Opposition Based Learning (OBL) initialization is the most well-known two-step method in EA literature [33], [34]. The definition of the opposite point is as follows:

Let $X_i(x_{i,1}, x_{i,2}, ..., x_{i,D})$ be the ith individual of the population and each variable $x_{i,j}$ be bounded by (a_i, b_j). The opposition point is defined as $\tilde{X}_i(\tilde{x}_{i,1}, \tilde{x}_{i,2}, ..., \tilde{x}_{i,D})$ while:

$$\tilde{x}_{i,j} = a_j + b_j - x_{i,j}, \quad j = 1, ..., D. \tag{2}$$

To produce a promising population, OBL method first generates a random population (let's call it original population) and then the opposition points of the original population are calculated based on Equation 2. Now, both populations are merged and the best individuals according to fitness function are selected to form the initial population for the EA (second step).

Another well-known two-step population initialization method is Quasi-opposition Based Learning (QBL). The QBL is not a hybrid version of QRS and OBL. Indeed, QBL is a modified version of OBL which tries to increase the population uniformity. Considering the definition of opposition points in Equation 2, the quasi-opposition point of X_i is $\breve{X}_i(\breve{x}_{i,1}, \breve{x}_{i,2}, ..., \breve{x}_{i,D})$ where:

$$\breve{x}_{i,j} = \begin{cases} rand(m_j, \tilde{x}_{i,j}) & \text{if } x_{i,j} \leq m_j \\ rand(\tilde{x}_{i,j}, m_j) & \text{if } x_{i,j} > m_j \end{cases} \tag{3}$$

where $m_j = \frac{b_j - a_j}{2}$ and $rand(\alpha, \beta)$ is a random number drawn uniformly from (α, β) range. Similar to OBL, after the calculation of quasi-opposition points, both original and quasi-opposition populations are merged into one big population. Then, fittest solutions are selected according to fitness function.

As described in [2], according to probability theory, in 50% of cases the distance between the opposition point and the unknown solution is less than the distance between the original point and the unknown solution. In [9], Rahnamayan et al. proved that points generated using QBL have more chances to be closer to unknown solutions than points produced by OBL.

Although OBL and QBL initialization methods achieved good results, they suffer from two problems; Firstly, both algorithms consume a part of computation budget to evaluate the fitness function. Secondly, since these methods calculate opposition and quasi-opposition points based on the original population, their performances to some extent depends on the quality of the original population.

TABLE II. CEC 2008 LSGO BENCHMARK FUNCTIONS

	Name of Function	Type of Function	Separability
$f1$	Shifted Sphere Function	Unimodal	Separable
$f2$	Shifted Schwefel's Problem 2.21	Unimodal	Nonseparable
$f3$	Shifted Rosenbrock's Function	Multimodal	Nonseparable
$f4$	Shifted Rastrigin's Function	Multimodal	Separable
$f5$	Griewank's Function	Multimodal	Nonseparable
$f6$	Shifted Ackley's Function	Multimodal	Separable

TABLE III. DE PARAMETER SETTINGS

Parameter	Value	Parameter	Value
Strategy	local to best/1/bin	Bound Constraints	None
Crossover Rate	0.90	Termination Criteria	Func. Eval.
F Weight	0.85	Max. Func. Eval.	5000 times Dim.

IV. EXPERIMENTAL SETUP

This study comprises two experimental parts. The first part is done on problems with 100, 500 and 1000 dimension sizes. We choose three different dimension sizes to assess the consequence of the initialization step on the final results of EAs in medium and large dimension sizes. In fact, we want to investigate whether a promising initialization method on 100 dimensional problems (as the medium dimension size) can also improve the EA outcomes in higher dimensions. In this part, population size is kept constant (i.e., 50) for all methods and problems.

In the second part, we examine problems with 500 dimension size using different population sizes. The aim of this part is to investigate whether increasing the population size can or cannot compensate the shortcomings of weak initialization methods. In other words, a promising initialization method should outperform RNG regardless of the chosen population size. Therefore in this part we compare advanced initialization methods with six different population sizes. Note that, apart from population size, other parameters are kept the same as the first part of the experiments.

To determine the effect of different initialization methods on LSGO problems, the CEC'2008 benchmarks are selected. The benchmark functions and their properties are presented in Table II. More detailed description of CEC'2008 benchmarks can be found in [35].

For experiments, a standard implementation of Differential Evolution (DE/local to best/1/bin) is used [36]. Among many EA models, DE is selected due to its simplicity and popularity in solving LSGO problems [16]. To keep it simple and the results reproducible by other researchers, common values for DE parameters are chosen. The input parameters and their values are presented in Table III. Note that since we only focus on initialization step, advanced DE variations such as those using self-adaptive parameter tuning [37] or cooperative coevolution versions [38] are avoided.

V. RESULTS AND DISCUSSIONS

The results of running the eight aforementioned initialization methods on CEC'2008 benchmark functions are presented in Tables IV, V and VI. These tables are obtained from the average values of 50 independent runs of DE on 100, 500 and 1000 dimensions. In all of these tables, methods which significantly outperformed RNG are shown with bold. Methods which their results are statistically similar to RNG are written

in italic. The best method for each function is also emphasized with an asterisk symbol (i.e, $*$) only if that method significantly enhanced RNG. Here, "significant" means Wilcoxon rank-sum test rejects the null hypothesis with at least 95% confidence. On the other hand, "statistically similar" means Wilcoxon rank-sum test do not rejects the null hypothesis. The null hypothesis in this test is that RNG results and its competitors' results represent the same statistical distribution.

Since this study aims to highlight the importance of considering more advanced population initialization methods as better alternatives to RNGs, in all statistical tests RNG is selected as the baseline method. Consequently, all initialization methods are only compared with RNG. Note that while other nonparametric statistical tests were available [39], Wilcoxon rank-sum test is selected due to its popularity in this context.

From Table IV, it is apparent that except OD, all other methods enhanced RNG in several cases. The results presented in this table indicates that all methods can solve $f5$ successfully and the performances of all methods on $f6$ are also statistically similar. In these cases, employing alternative methods has no considerable advantages. In the other cases (i.e. $f1$, $f2$, $f3$ and $f4$), however, using the advanced methods is notably advantageous. This table also shows that most of the methods (e.g., TNT, SBL, GLP, OD, QBL) can be used as the risk-free alternatives of RNG because their results are significantly better or statistically similar to RNG.

Situations in Table V and VI are more or less similar to Table IV. Although there are no significant improvements for $f3$ in 500 and 1000 dimensions and $f4$ in 500 dimension, advanced methods considerably boost DE final results of remaining functions (i.e., $f1$, $f2$, $f5$ and $f6$). These three tables confirm that even in very hard LSGO problems (i.e., non-separable, multi-modal and large scale), initialization methods can significantly influence EA performance. Indeed, these results reveal RNG should not be considered as the first choice of population initialization when dealing with LSGO problems.

Table VII is presented for a more clear comparison between the performance of advanced initialization methods on problems with different dimension sizes. For each advanced method and dimension size, a triple like α-β-γ is given. Here, α denotes the number of functions which are significantly improved by the advanced method. β represents the number of functions which their results are statistically similar to RNG results and γ shows how many times that advanced method significantly worse than RNG (according to aforementioned Wilcoxon rank-sum statistical test). To shed more light on this matter, consider OBL in 100 dimension. In this case the triple is 3 - 2 - 1. From this triple we know that OBL significantly surpassed RNG on three functions, statistically similar to RNG on two functions, and in one function, OBL produces results which are significantly worse than RNG results.

Table VII is quite revealing in several ways. First, comparing each method's performance in different dimension sizes indicates that the rank of advanced methods (except OD) are very consistent. For instance, TNT significantly outperforms RNG in at least 50% of functions in every dimension sizes while GLP is never significantly surpassed by RNG in any dimension size. Another striking observation is that, regardless of dimension size, TNT, SBL, OBL and QBL can significantly enhance RNG in at least 50% of functions. QBL is also found to be the best method among these eight initialization methods while both GLP and QBL can be considered as the risk-free substitute of RNG on LSGO problems (i.e., never be significantly worse than RNG).

Having discussed the effects of advanced initialization methods on LSGO problems with different dimension sizes, now we want to analyse the performance of these methods when population sizes are varied. Figure 1 illustrates the median values of 50 independent runs of all eight initialization methods with six different population sizes (i.e., from 50 to 300 with 50 step size).

It can be seen in Figure 1 that, except for GLP, the performance of DE decreases when population size is increased. This result is expected because computational budget is kept fixed for all population sizes. This means when population size is increased, the number of iterations (i.e. maximum generation number) is reduced and DE may not fully converge.

Apart from adverse consequence of population size increment, Figure 1 reveals that the ranks of methods are strongly consistent when population size is changed. UD and OD, for example, are almost always the worst methods while TNT and QBL are always one of the best methods. Among all initialization methods, GLP performance fluctuates the most. This unique behaviour of GLP demands more experimental and analytical studies.

Table VIII is also presented to support the findings of Figure 1. The triples in this table are produced in the same way as triples in Table VII. As it is apparent in Table VIII, a number of advanced methods (e.g. TNT, SBL, OBL and QBL) considerably outperform RNG in the most cases (in at least 83% of cases). This means that regardless of population size, RNG is far from the best choice for population initialization when dealing with LSGO problems.

Another valuable observation from Table VIII and Figure 1 is that there is no improvement (i.e. fitness value decrement) for RNG associated with population size increment. In other words, increasing population size cannot remedy RNG weakness. Increasing population size, indeed, not only adversely affect RNG performance in all cases, it amplifies the gap between RNG and other promising initialization methods. For a clear evidence, consider how many times each method surpasses RNG when population size is 50 (see the first numbers of each triple in the first row after the header in Table VIII). Then, compare these values with the numbers when population size is 300 (see sixth row after the header of the table). It is obvious that RNG is beaten more times when the population size is 300 (in comparison with population size of 50).

This part of our experiment concludes that RNG, even with very large population size, is not a proper choice for population initialization when dealing with LSGO problems. TNT, QBL, SBL and GLP, however, can be seen as the better alternatives among all methods that have been examined in this study.

Another observation from both parts of the experiments is that none of the initialization categories is significantly better than the others. Whilst RNG, for example, do not perform well on LSGO, TNT which is from the same category as

TABLE IV. MEAN OF 50 INDEPENDENT RUNS (DIMENSION SIZE = 100, POPULATION SIZE = 50)
STARED*, **BOLD** OR *italic* NUMBERS DENOTE BEST METHODS, SIGNIFICANTLY BETTER RESULTS OR STATISTICALLY SIMILAR RESULTS, RESPECTIVELY

	RNG	TNT	SBL	GLP	UD	OD	OBL	QBL
$f1$	7.492270e-21	**4.265986e-21**	**6.803830e-21**	**6.148537e-21**	**7.415238e-21**	*5.295613e-21*	**5.972895e-21**	**4.041908e-21***
$f2$	7.737881e+00	**4.543428e+00***	**6.682139e+00**	*6.347901e+00*	7.776422e+00	*5.575648e+00*	**7.578740e+00**	**6.318157e+00**
$f3$	1.031083e+02	*9.638965e+01*	*9.949269e+01*	*9.950065e+01*	1.039764e+02	*9.778514e+01*	1.050150e+02	**9.731273e+01**
$f4$	2.553259e+02	*2.564297e+02*	*2.598275e+02*	*2.587925e+02*	*2.565291e+02*	*2.589391e+02*	*2.607131e+02*	*2.584029e+02*
$f5$	0.000000e+00	*0.000000e+00*	*0.000000e+00*	*0.000000e+00*	*0.000000e+00*	*0.000000e+00*	*0.000000e+00*	*0.000000e+00*
$f6$	1.697432e+01	**1.709277e-09**	**2.780043e-11**	**4.727463e-11**	1.985263e+01	*1.112577e-11*	**1.139093e-10**	**9.184209e-12***

TABLE V. MEAN OF 50 INDEPENDENT RUNS (DIMENSION SIZE = 500, POPULATION SIZE = 50)
STARED*, **BOLD** OR *italic* NUMBERS DENOTE BEST METHODS, SIGNIFICANTLY BETTER RESULTS OR STATISTICALLY SIMILAR RESULTS, RESPECTIVELY

	RNG	TNT	SBL	GLP	UD	OD	OBL	QBL
$f1$	2.602724e-08	**1.096963e-08***	**2.335575e-08**	*2.363324e-08*	2.875633e-08	3.700760e-08	**2.547586e-08**	**1.567035e-08**
$f2$	8.283352e+01	**7.570587e+01**	*8.079759e+01*	*8.002004e+01*	**7.989864e+01**	*7.368558e+01*	**7.974286e+01**	**7.834218e+01**
$f3$	6.635788e+02	*6.531711e+02*	*6.525941e+02*	*6.732033e+02*	*6.823002e+02*	*6.550677e+02*	*6.952809e+02*	*6.692513e+02*
$f4$	2.535056e+03	*2.632127e+03*	*2.548903e+03*	*2.558774e+03*	*2.573831e+03*	2.559681e+03	2.549602e+03	*2.556631e+03*
$f5$	3.223598e-09	**1.253168e-09***	**2.857621e-09**	*2.751713e-09*	3.573696e-09	4.651590e-09	**3.031251e-09**	**1.844874e-09**
$f6$	9.651435e+00	**9.154128e-06**	**1.023049e+00**	**2.762930e+00**	1.333887e+01	2.108006e+01	**5.804797e+00**	**7.428489e-06***

TABLE VI. MEAN OF 50 INDEPENDENT RUNS (DIMENSION SIZE = 1000, POPULATION SIZE = 50)
STARED*, **BOLD** OR *italic* NUMBERS DENOTE BEST METHODS, SIGNIFICANTLY BETTER RESULTS OR STATISTICALLY SIMILAR RESULTS, RESPECTIVELY

	RNG	TNT	SBL	GLP	UD	OD	OBL	QBL
$f1$	1.786221e-09	**1.088676e-09***	**1.572158e-09**	*1.586478e-09*	2.767689e-09	2.365400e-09	**1.750659e-09**	**1.238254e-09**
$f2$	1.175865e+02	**9.502366e+01**	**9.482039e+01***	*9.489372e+01*	1.217340e+02	**1.014329e+02**	**1.158858e+02**	**1.148139e+02**
$f3$	2.286005e+03	*2.329177e+03*	*2.256663e+03*	*2.370488e+03*	*2.313513e+03*	*2.223574e+03*	*2.289505e+03*	*2.316646e+03*
$f4$	7.179364e+02	1.235047e+03	7.587876e+02	*7.460613e+02*	1.064208e+03	**7.168448e+02***	7.291611e+02	*7.919397e+02*
$f5$	1.176424e-10	*8.338685e-11*	*1.123224e-10*	*1.101994e-10*	1.809505e-10	1.604613e-10	1.184557e-10	**8.113715e-11***
$f6$	7.407606e-01	**2.804952e-06**	**3.747755e-05**	**1.300744e-05**	3.814163e+00	2.116551e+01	**1.543886e-05**	**1.971337e-06***

TABLE VII. PERFORMANCE COMPARISON OF ADVANCED INITIALIZATION METHODS ON PROBLEMS WITH DIFFERENT DIMENSION SIZES (POPULATION SIZE = 50)
NUMBERS IN EACH TRIPLE DENOTE THAT METHOD ON HOW MANY FUNCTIONS IS SIGNIFICANTLY BETTER THAN RNG, STATISTICALLY SIMILAR TO RNG OR SIGNIFICANTLY WORSE THAN RNG, RESPECTIVELY

Dimension	TNT	SBL	GLP	UD	OD	OBL	QBL
100	3 - 3 - 0	3 - 3 - 0	2 - 4 - 0	1 - 2 - 3	0 - 6 - 0	3 - 2 - 1	4 - 2 - 0
500	4 - 2 - 0	3 - 3 - 0	1 - 5 - 0	1 - 2 - 3	0 - 2 - 4	4 - 1 - 1	4 - 2 - 0
1000	3 - 2 - 1	3 - 2 - 1	1 - 5 - 0	0 - 1 - 5	2 - 1 - 3	3 - 1 - 2	4 - 2 - 0
Summation	10 - 7 - 1	9 - 8 - 1	4 - 14 - 0	2 - 5 - 11	2- 9 - 7	10 - 4 - 4	**12 - 6 - 0**
Percentage	56% - 38% - 6%	50% - 44% - 6%	22% - 78% - 0	11% - 28% - 61%	11%- 50% - 39%	56% - 22% - 22%	**67% - 33% - 0%**

RNG, is ranked as one of the best alternatives (for more evidence compare OBL and QBL in the last rows of Tables VII and VIII). This fact suggests further investigation about the reason behind the good/bad performance of these methods is needed.

VI. CONCLUSION

This study investigated the effect of advanced population initialization methods for EAs when dealing with high-dimensional problems. One of the major findings is that basic random number generators should not be advocated for population initialization in EAs. In fact, there exist other advanced methods, which, regardless of the dimension size, can significantly improve the performance of EAs. It is also shown that increasing the population size while fixing the computational budget cannot improve the performance of using basic random number generators. Indeed, some methods such as chaotic numbers, low-discrepancy sequences and quasi opposition-based methods can improve the performance of EAs no matter how large the population size is. Another observation from this study is that in all categories of initialization methods, we can observe both promising and weak methods. In other words, all categories of initialization methods are worth being further investigated.

The findings from this study contribute to the research community. First, it recommends to employ more advanced initialization methods when dealing with large-scale optimization problems using EAs. Secondly, instead of blindly increasing the population size, it is better to considere using more promising initialization methods. In fact, further investigation still needs to be done to discover why some initialization methods perform significantly better than the others.

ACKNOWLEDGEMENT

This study was supported by NSFC under Grant No. 61005051, SRFDP under Grant No. 20100092120027 and ARC Discovery Grant (DP120102205). The authors would like to thank Dr. Ke Tang for providing valuable resources.

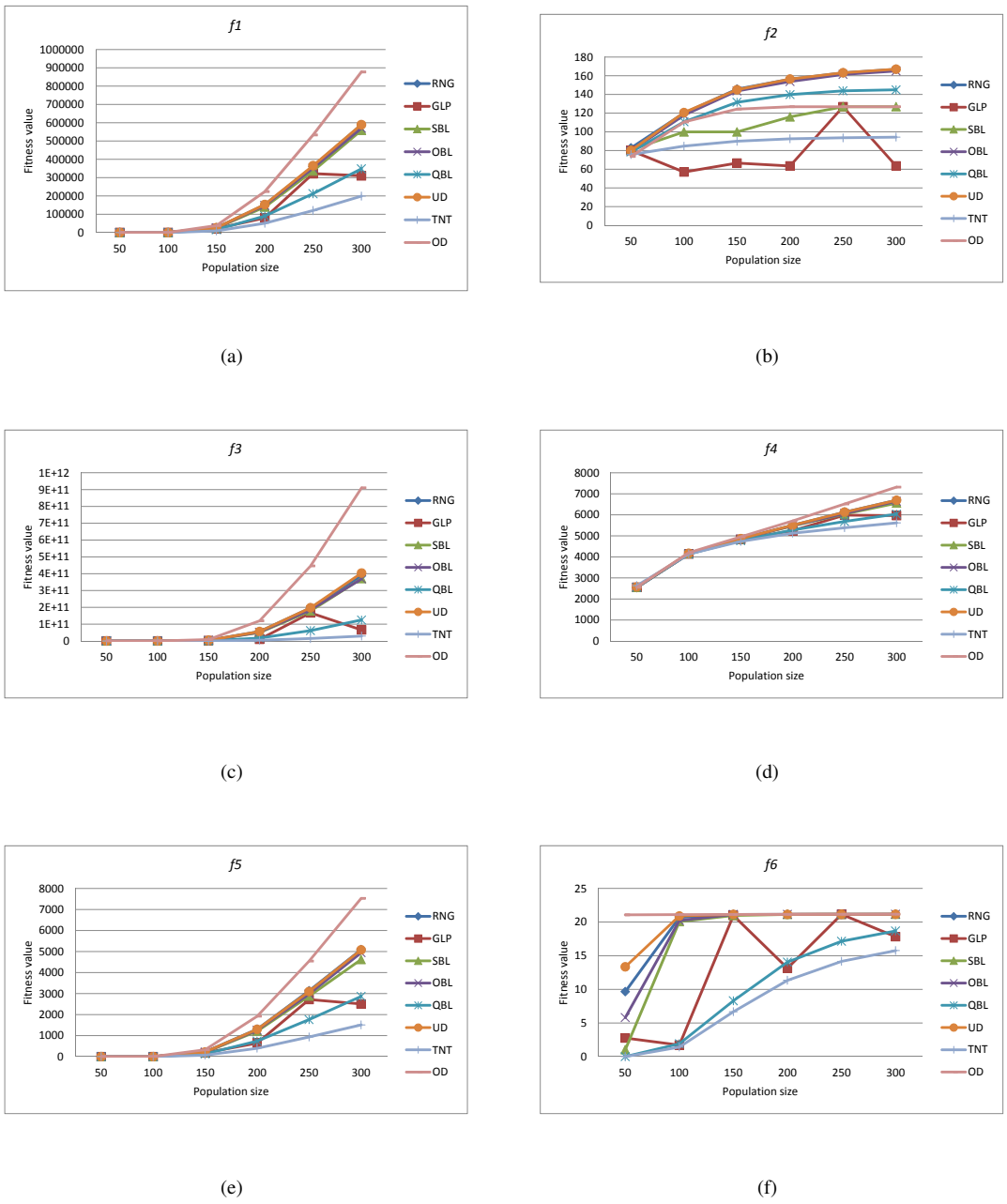

Fig. 1. Comparison between advanced initialization methods with different population sizes on CEC'2008 LSGO benchmark functions(dimension size = 500)

TABLE VIII. PERFORMANCE COMPARISON OF ADVANCED INITIALIZATION METHODS WITH DIFFERENT POPULATION SIZES (DIMENSION SIZE = 500) NUMBERS IN EACH TRIPLE DENOTE THAT METHOD ON HOW MANY FUNCTIONS IS SIGNIFICANTLY BETTER THAN RNG, STATISTICALLY SIMILAR TO RNG OR SIGNIFICANTLY WORSE THAN RNG, RESPECTIVELY

Population	TNT	SBL	GLP	UD	OD	OBL	QBL
50	4 - 2 - 0	3 - 3 - 0	1 - 5 - 0	1 - 2 - 3	0 - 2 - 4	4 - 1 - 1	4 - 2 - 0
100	6 - 0 - 0	5 - 1 - 0	4 - 2 - 0	0 - 0 - 6	0 - 4 - 2	6 - 0 - 0	6 - 0 - 0
150	6 - 0 - 0	6 - 0 - 0	4 - 2 - 0	3 - 0 - 3	0 - 5 - 1	5 - 0 - 1	5 - 1 - 0
200	6 - 0 - 0	6 - 0 - 0	4 - 2 - 0	2 - 0 - 4	0 - 6 - 0	5 - 0 - 1	5 - 1 - 0
250	6 - 0 - 0	5 - 1 - 0	5 - 1 - 0	2 - 0 - 4	0 - 6 - 0	6 - 0 - 0	5 - 1 - 0
300	6 - 0 - 0	5 - 1 - 0	5 - 1 - 0	3 - 0 - 3	0 - 5 - 1	6 - 0 - 0	6 - 0 - 0
Summation	**34 - 2 - 0**	30 - 6 - 0	23 - 13 - 0	11 - 2 - 23	0 - 28 - 8	32 - 1 - 2	31 - 5 - 0
Percentage	**94% - 6% - 0%**	83% - 17% - 0%	64% - 36% - 0%	30% - 6% - 64%	0%- 78% - 22%	89% - 3% - 8%	86% - 14% - 0%

REFERENCES

[1] M. Clerc, "Initialisations for particle swarm optimisation," Tech. Rep., 2008.

[2] 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.

[3] 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.

[4] 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*, 2012.

[5] 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.

[6] W.-f. Gao and S.-y. Liu, "A modified artificial bee colony algorithm," *Computers & Operations Research*, vol. 39, no. 3, pp. 687–697, 2012.

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

[8] L. Peng, Y. Wang, G. Dai, and Z. Cao, "A novel differential evolution with uniform design for continuous global optimization," *Journal of Computers*, vol. 7, no. 1, pp. 3–10, 2012.

[9] 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.

[10] H. Wang, Z. Wu, J. Wang, X. Dong, S. Yu, and C. Chen, "A new population initialization method based on space transformation search," in *Natural Computation, 2009. ICNC'09. Fifth International Conference on*, vol. 5. IEEE, 2009, pp. 332–336.

[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*, 2012.

[12] 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.

[13] R. Morrison, "Dispersion-based population initialization," in *Genetic and Evolutionary ComputationGECCO 2003*. Springer, 2003, pp. 204–204.

[14] 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.

[15] L.-Y. Tseng and C. Chen, "Multiple trajectory search for large scale global optimization," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, 2008, pp. 3052–3059.

[16] M. N. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 2010, pp. 1–8.

[17] 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.

[18] M. Richards and D. Ventura, "Choosing a starting configuration for particle swarm optimization," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 3. IEEE, 2004, pp. 2309–2312.

[19] 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.

[20] C. Yanguang, M. Zhang, and C. Hao, "A hybrid chaotic quantum evolutionary algorithm," in *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, vol. 2. IEEE, 2010, pp. 771–776.

[21] M. Clerc, "Randomness matters," 2012.

[22] I. H. Sloan, "Lattice methods for multiple integration," *Journal of Computational and Applied Mathematics*, vol. 12, pp. 131–143, 1985.

[23] 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.

[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] N. Q. Uy, N. X. Hoai, R. McKay, and P. M. Tuan, "Initialising pso with randomised low-discrepancy sequences: the comparative results," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE, 2007, pp. 1985–1992.

[26] L. Peng and Y. Wang, "Differential evolution using uniform-quasi-opposition for initializing the population," *Information Technology Journal*, vol. 9, no. 8, pp. 1629–1634, 2010.

[27] W.-f. Gao, S.-y. Liu, and L.-l. Huang, "Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique," *Communications in Nonlinear Science and Numerical Simulation*, 2012.

[28] A. Gutierrez, 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.

[29] C.-H. Chou and J.-N. Chen, "Genetic algorithms: initialization schemes and genes extraction," in *Fuzzy Systems, 2000. FUZZ IEEE 2000. The Ninth IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 965–968.

[30] R. A. Khanum and M. A. Jan, "Centroid-based initialized jade for global optimization," in *Computer Science and Electronic Engineering Conference (CEEC), 2011 3rd*. IEEE, 2011, pp. 115–120.

[31] 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.

[32] H. Maaranen, K. Miettinen, and A. Penttinen, "On initial populations of a genetic algorithm for continuous optimization problems," *Journal of Global Optimization*, vol. 37, no. 3, pp. 405–436, 2007.

[33] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Opposition-based differential evolution for optimization of noisy problems," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. IEEE, 2006, pp. 1865–1872.

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

[35] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, and Z. Yang, "Benchmark functions for the cec'2008 special session and competition on large scale global optimization," *Nature Inspired Computation and Applications Laboratory, USTC, China*, 2007.

[36] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution*. Springer, 1997.

[37] 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.

[38] M. N. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 2010, pp. 1–8.

[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.