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

A Review of Population Initialization Techniques for Evolutionary Algorithms

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— **Although various population initialization techniques have been employed in evolutionary algorithms (EAs), there lacks a comprehensive survey on this research topic. To fill this gap and attract more attentions from EA researchers to this crucial yet less explored area, we conduct a systematic review of the existing population initialization techniques. Specifically, we categorize initialization techniques from three exclusive perspectives, i.e., randomness, compositionality and generality. Characteristics of the techniques belonging to each category are carefully analysed to further lead to several sub-categories. We also discuss several open issues related to this research topic, which demands further in-depth investigations.**

I. INTRODUCTION

Evolutionary algorithms (EAs) are typically a population-based stochastic search technique, which share one common algorithmic step: population initialization. The role of this step is to provide an initial guess of solutions. Then, these initially guessed solutions will be iteratively improved in the course of the optimization process until a stopping criterion is met. Generally, *good* initial guesses can facilitate EAs to locate the optima [1], [2]. On the contrary, starting from *bad* guesses may prevent EAs from finding the optima [3]. This issue becomes more serious when solving large-scale optimization problems using a population of finite size [4].

In black-box optimization (which EAs are apt to deal with [5]), there exists no prior information about the search landscape of a given problem [6]. Therefore, *good* and *bad* initial populations cannot be determined. In such a case, EA researchers often employ pseudo-random number generators (PRNGs) to produce the initial population [7]. The rationale behind this is that PRNGs can generate uniformly distributed samples [8] and thus a population initialized using PRNGs tends to cover promising regions (containing global optima or good local optima) of the search space [3], [6], [9].

Since the population size is always limited, the chance for a population to cover promising regions of the search space decreases as increasing the dimension of the search space. This fact becomes obvious when dealing with large-scale optimization problems [4], [10]. Recently, research community has started to study the effects of other initialization techniques on the performance of EAs [11], [12]. These investigations revealed that many promising alternatives are available to be used as population initializers for EAs. For example, some studies had claimed that advanced initialization techniques can increase the probability of finding global optima [7], reduce the variation of final searching results [13], decrease the computational costs [7] and improve the solution quality [8]. Based on these findings, a large and growing body of literatures has been devoted to study the new ways of population initialization for EAs [11].

While there exists a considerable number of publications regarding population initialization techniques, little attention has been paid to summarize and analyse them in a comprehensive and systematic way. To our best knowledge, [11] is the only attempt ever made to provide a brief review of some existing population initialization techniques. The current paper, however, further expands the previous work in several ways. Firstly, we provide a comprehensive survey by including more initialization techniques, especially some very recent techniques not mentioned in [11]. Secondly, we redefine the categorization of population initialization techniques in a clear, concise and systematic manner. Thirdly, we discuss the trends and open questions in this research topic and provide some guidelines for the future research. In fact, this study aims to highlight the importance and challenges of the research of population initialization for EAs. It will help EA researchers to understand the whole picture of the current research in this topic, and facilitate them to choose suitable population initialization techniques in their research.

The remaining of this paper is organized as follows. Next section introduces the new categorization. Sections III, IV and V describe three general categories in detail and discuss their involved sub-categories with representative techniques. The limitations in the existing works, some open questions as well as guidelines for future studies are provided in Section VI. Finally, Section VII concludes the paper.

II. CATEGORIZATION

Population initialization techniques have various types of different characteristics. Previously, a very few categorization works had been done to group these techniques into different categories [11]. Although the existing categorization works are informative, they suffer from two major problems. Firstly, most of them are not comprehensive enough to cover all types of existing techniques. Secondly, they mostly categorize the techniques from limited perspectives, e.g., randomness, without considering other important factors that can exclusively characterize the techniques.

In this paper, we propose the new categorization which covers all of the existing population initialization techniques. The proposed categorization groups the existing techniques from three exclusive perspectives that are easy-to-understand

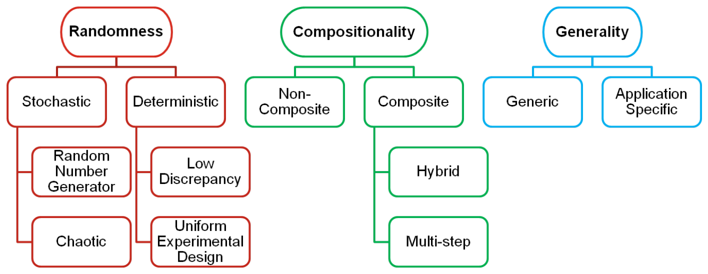

Fig. 1. Three categorizations of population initialization techniques, based on randomness, compositionality and generality.

for general users. These aspects are: *randomness*, *compositionality* and *generality*. Within each of these three categories, we further demarcate representative techniques into several sub-categories and describe their properties in details. The hierarchy of the proposed categorization is illustrated in Figure 1.

As shown in Figure 1, three categories are further demarcated into several sub-categories according to more specific criteria, and some of these sub-categories are further refined.

The three categorization criteria are determined based on two facts. Firstly, each criterion describes the technique from a unique and independent aspect. For example, whether a technique is random or not does not depend on whether it is compositional and/or general. Therefore, we can describe a technique from exclusive perspectives. Secondly, each criterion is easy-to-understand for both EA researchers and practitioners. For example, if an EA researcher wants to choose an appropriate population initializer for a multi-start EA, the randomness of a population initializer must be taken into account. As another example, the generality of a technique plays an important role when a practitioner needs to find a proper initialization technique suitable for solving a specific problem (see Section V for more information).

The following sections detail each of the three categories. Their sub-categories with the pros and cons of the representative techniques also being discussed.

III. Randomness

From the aspect of randomness, a sequence of numbers can be seen as *completely deterministic* to *truly random* [14]. While there is no agreement on a universal definition for randomness [15], a true random sequence is usually described as a sequence having strong properties such as complete unpredictability, incompressibility and irregularity [16]. Although several tools are available to measure these properties [17], [18], it is impossible to prove that the given sequence is truly random [19]. In turn, these tools can be employed as tests to determine that the sequence is not truly random (if it cannot pass the tests) [19]. Some of these tools may also be useful to see if a given sequence is *computationally random* or *statistically random* [8].

Obviously, the results of aforementioned tests (on unpredictability, for example) are very sensitive to the power of the adopted tools [14]. To avoid such confusion, a simple, yet stable, alternative procedure is proposed here. In this paper, initialization techniques are categorised only according to their dependency on initial seeds. In other words, an initializer is considered as *stochastic* if it generates different populations while it is fed by different initial seeds. In contrast, techniques which consistently produce exactly the same population, regardless of any initial seed, are considered as *deterministic*. Note that stochastic and deterministic are the attributes of the resulted populations, not the generating algorithms.

A. Stochastic Techniques

As discussed above, population initialization techniques where their results depend on initial seeds, are labelled as stochastic initializers. In this paper, we assume that the initial seed, which is provided by an external random source, is the only cause of randomness. The group of stochastic techniques can be divided into two subgroups: *Pseudo-Random Number Generators* and *Chaotic Number Generators*. Following paragraphs provide more details regarding these two subgroups.

1) Pseudo-Random Number Generator (PRNG): Due to the disability of deterministic machines (i.e., digital computers) in producing true random numbers [17], and also the lack of efficient techniques to sample random numbers from physical phenomena (e.g., radioactive decay [20] or atmospheric noise [21]), PRNGs are widely used in many applications to generate numbers which look like random [14].

Generally speaking, PRNGs can be ranked based on two key factors: *cycle time* (a.k.a. period length) and equidistribution. In literature, cycle time is defined as the smallest integer that a PRNG repeats producing previously produced numbers; and equidistribution means all points in the range have equal frequency or probability of occurrence [14]. PRNGs which pass some tests (e.g., DieHard [17], [18] and TestU01 [22]) can be considered as computationally or statistically random number generators.

In EA literature, PRNGs are known as the most commonly used population initializers [23], [24]. Among many PRNG variants, WELL [25], KISS [26] and Mersenne Twister [27] are the most widely used PRNG algorithms in the EA domain. The main properties which make them very common are *simplicity* and *uniformity*. Since fast PRNG tools are available in every programming language and there is no restriction on the number of points (i.e., population size) and dimension size (i.e., number of decision variables), they can be easily applied to every problem. Moreover, where the dimensionality of the problem is not very high and population size is large enough, PRNGs can provide initial populations with satisfactory level of uniformity [8]. As mentioned earlier, using initial population with a high level of uniformity can decrease the chance of missing a considerable part of search space through optimization process.

Uniform populations generated by PRNGs can be easily transformed to biased populations. Here, biased means the points in the population are not evenly distributed. In fact, they are scattered according to other distributions such as normal or Gaussian distribution. Some previous works prefer to use biased randomly generated points as initial population [2], [8], [10].

Apart from the interesting properties of PRNGs, they suffer from the *curse of dimensionality* [3]. Indeed, PRNGs cannot produce perfect evenly distributed points [4], [10]. This drawback gets worse when the search space dimensionality grows or/and the population size decreases [11].

To lessen the adverse effect of dimensionality on the performance of PRNGs, one may propose to increase the population size. However, as empirically shown in [11], increasing population size while computational budget is fixed cannot remedy the issue. In fact, blindly increasing the population size may result to early termination which in turn may cause the population not to converge at all. This can be even worse than a converged algorithm which missed a considerable portion of the search space due to poor uniformity of the initial population [11]. Consequently, assuming uniformity as a key factor of initial population, EAs need more advanced techniques to provide better uniformly distributed initial populations.

2) Chaotic Number Generator (CNG): Beside PRNGs, chaotic techniques are also employed to produce stochastic initial populations [28], [29], [30], [31], [32]. Technically, *Chaos theory* studies the behaviour of dynamical systems which are very sensitive to their initial conditions. Ergodicity (i.e., the ability to traverse all states in a certain region [28]), randomness and regularity are the main properties of chaotic systems [33]. Since these properties are desirable in many applications, lots of CNGs are proposed and widely used [34], [35]. In order to produce a chaotic population, a proper map is required. In a very general form, one-dimensional chaotic maps work as follows:

$$x_{i,j}^{k+1} = f_\mu(x_{i,j}^k) \tag{1}$$

where $x_{i,j}^k$ is j^{th} variable of i^{th} individual in k^{th} iteration and μ is the set of user defined parameters. Generally $x_{i,j}^0$ is chosen randomly and successive points are produced by iteratively applying the chaotic map.

To the best of our knowledge, no study has been done to compare the uniformity of PRNGs' and CNGs' generated populations. However, it has been shown that adopting chaotic initialization techniques can improve performance of EAs in terms of population diversity, success rate and convergence speed [30], [32], [36].

Apart from the advantages of CNGs, they suffer from a number of disadvantages. Firstly, most of previously proposed chaotic maps are designed for one, two or three dimensional spaces [37]. This means that the interesting properties of chaos may be visible in those low dimensional projections, but it can hardly be generalized to higher dimensions. More studies are required to investigate the performance of high dimensional populations which are generated using low dimensional maps.

Secondly, the behaviour of CNGs are very sensitive to the initial condition and its parameter settings [36], [37]. For example, in Tent map with $\mu < 1$, the resulting population will converge towards 0 regardless of the initial seed. For $1 < \mu < 2$, however, all values close to 0 or $\mu/(\mu+1)$ move away from them rapidly (but still remain in range $[0,1]$). On the other hand, when $\mu > 2$ almost every point in range $[0,1]$ eventually diverge towards infinity. For Tent map, the only proper value for μ which produces chaotic sequences is 2 [28].

Thirdly, the performance of CNGs is very sensitive to precision of their implementations. As impractically studied in [38], different precision levels cause chaotic sequences with different periodicities which can significantly affect EAs performance.

Fourthly, existence of some attractors may cause the population to converge to a few fixed points. In the case of the logistic map with parameter $r = 4$ and an initial seed in range $(0,1)$, for example, 0, 0.25, 0.5 and 0.75 are known as strong attractors [39]. The population must be checked against these attractors; otherwise, it may converge towards 0 after a number of iterations (depending on the precision).

Finally, it is not clear yet why in some cases a few maps perform considerably better than the others [37]. While the reason behind CNGs' performance is not investigated, general practitioners may face difficulties finding the best maps (along with best parameter settings) for their particular problems.

B. Deterministic Techniques

As mentioned earlier, techniques which always generate the same population (regardless of the initial seed) are known as deterministic. In contrast with the stochastic techniques, randomness and unpredictability are not important objectives here [40]. In turn, deterministic initializers are specially designed to provide evenly distributed points in the entire search space [11]. Recently, these techniques attract more attention because in the absence of prior knowledge about the problem, uniformity of the initial population can enhance the exploration ability of EAs in early iterations [6]. This may result in converging to a better solution (in terms of objective value) and saving a considerable amount of computational budget [3], [8].

In literature, deterministic point generators are also referred to as *low-discrepancy* techniques [40]. Literally, discrepancy means non-uniformity and hence discrepancy measures are tools for determining non-uniformity level of a given point set [3], [40]. In other words, point sets with low discrepancy are those with high level of uniformity. Generally, two slightly different approaches for generating low-discrepancy sets are proposed so far: *quasi-random sequence* and *uniform experimental design* [14].

1) Quasi-Random Sequence (QRS): The term "random" in the name of QRS should not confuse readers. These sequences are neither true random nor pseudo-random. Indeed, QRSs in the original form are completely deterministic and no random element (e.g., random initial seed) is involved in their algorithms [9].

Seeking low-discrepancy sequences, QRS techniques have the support of theoretical upper-bounds on discrepancy of the resulting sequences. Technically, when the population size is large enough (i.e., $n \to \infty$), these limits show how much a particular QRS can be non-uniform in the worst case scenario [40]. Knowing these limits, QRS techniques try to find the optimal parameters to decrease the upper-bounds or to approach the lower-bounds [14]. Assuming positive correlation between the discrepancy of initial population and the objective value of the final solution (in minimization problems), one can select a proper QRS technique (with the least discrepancy) prior running EA. Since discrepancy calculation is usually easier than running several EA runs (in terms of computational complexity), having such theoretical limits is a worthy bonus for QRSs in comparison with the others.

Although QRS techniques have strong theoretical advantages over stochastic (and also other deterministic) techniques, they suffer from some limitations. Firstly, the theoretical bounds on discrepancy may not be very beneficial in practice due to unsatisfied assumptions. In high dimensional spaces, for example, population size is relatively smaller than what it should be to satisfy the underlying assumptions [11]. Indeed, some previous studies raised doubt about QRS techniques having such superiority (in terms of discrepancy) over PRNGs in high dimensions [41].

Secondly, various numerical algorithms for measuring discrepancy of a given sequence have been proposed [42]. These measures in some cases contradict each other. This means, a sequence may look more uniform than another sequence according to some discrepancy measures but less uniform in terms of other discrepancy measures. This contradiction in discrepancy measures makes it difficult for general practitioners to compare QRSs in order to find the best technique prior running the entire EA process.

Finally, any correlation between discrepancies and solution's objective values has not been proven yet. Consequently, even finding a QRS initializer with the least discrepancy values might not result in the best final objective value after running EA. These shortcomings can be the reasons behind the unpopularity of QRS in high dimensional optimization.

2) Uniform Experimental Design (UED): UED is a kind of space-filling algorithm which looks for points to be evenly scattered in a given range. Since its first introduction in 1980, UED has been widely used in industrial and computer simulation designs [43]. For some low-dimensional spaces, UED tables have been calculated and published.

Suppose we seek a complete grid in a D dimensional space which each variable has exactly q different values (i.e., levels). Then, the total number of points in the grid (i.e., population size) would be q^D. In theory, having large enough q results in a perfectly uniform population. However, evaluating such big population is practically impossible even for small scale problems. To lessen this difficulty, UEDs can be used to systematically select a smaller number of points from the complete grid which is still uniform. So far, a wide range of UEDs such as *uniform design* [44] and *orthogonal design* [45] has been employed as EA population initializer.

In comparison with QRSs, UEDs have two main advantages over them. Firstly, QRSs only consider one-dimensional projection uniformity; while non-orthogonal and orthogonal UEDs consider two and D-dimensional (in addition to one-dimensional) projection uniformities [43]. These extra considerations can provide more desirable regularity and uniformity. Secondly, UEDs generally generate discrete points while QRSs are originally designed for real-value spaces. This property helps UEDs to be directly applicable to nominal and discrete optimization problems.

Obviously, UEDs have some limitations. Firstly, the performance of many UEDs depends on the parameter settings. In orthogonal design (OD) [46], for example, the number of levels (i.e., q) plays a very important role. While large values of q can result in more uniform population, they can increase population size exponentially. This extremely large population size prevents users from using OD directly on moderate or large scale problems. To remedy this problem, [44] suggests to evaluate all generated points and then pick the best subset according to the objective values. This solution potentially wastes a considerable portion of computational budget in the early stage while it could be used in the course of optimization. In contrast, [45] suggests to group variables using some heuristics and use the same values for all variables in each group. This solution can reduce the resulting population size, but seriously affect the uniformity of the population. Moreover, variable grouping forces extra computational costs to practitioners.

IV. Compositionality

In this paper, compositionality is defined as the number of standalone procedures that are involved in a technique. According to this criterion, population initialization techniques fall into *composite* and *non-composite* groups. Following paragraphs provide more details regarding these two groups.

A. Non-compositional

From the compositionality point of view, all basic techniques which produce populations in only one single step are labelled as non-compositional. Hence, regardless of being stochastic, deterministic, generic or application specific, as long as a technique cannot be divided into disjoint population initialization techniques, it is considered as a non-compositional technique. Therefore, all techniques which were reviewed in Section III are non-compositional unless one hybridizes them.

B. Compositional

In contrast with non-compositional group, techniques which comprise more than one stage are labelled as compositional. Two subgroups of compositional techniques are previously proposed in literature: *hybrid* and *multi-step* techniques [11]. Each component of a hybrid technique can be separately applied as a non-compositional technique. For example, while CNG and PRNG techniques can be separately employed as individual population initializers, one may use a CNG to generate the initial seed for a PRNG.

Another example of hybrid initialization techniques which have been used in several studies are those that try to bring some randomness to QRS techniques. This way, the resulting population may have both uniformity of QRS population and randomness of PRNGs. Based on the employed hybridization techniques, the resulting algorithms may be different in name and characteristics. The *random start* QRS [40], *scrambled* QRS [7], [47], [48] or *mixed pseudo-quasi-random sequence* [14] are some examples of this category of initializations.

In general, hybrid techniques theoretically inherit the advantages and disadvantages of the basic techniques which they are made from [11]. Consequently, studying the basic components can shed more light on hybrid techniques as well. On the other hand, when our knowledge about the basic components (i.e., non-compositional techniques) is insufficient, studying hybrid techniques would provide little benefit and interest.

As opposed to hybrid techniques, a multi-step technique comprises of two or more components which at least one of them cannot be employed as an standalone initializer. In other words, multi-step techniques generally process and refine the previously generated population in later steps. One of the most popular multi-step techniques which is widely used in different algorithms and applications is the family of *opposition based learning* (OBL) techniques [24], [31], [49], [50].

In the first step, OBL techniques generate a set of points called original population. Original population can be generated using any initializer technique (e.g., PRNG [49], CNG [31] or UED [51]). Then, some simple heuristic rules are employed to produce another population with the same size in the second step. This new population is generally referred to as the opposite population. Finally, a subset of the union of both populations is selected based on their fitness values. Equation 2 shows the heuristic rule which produces an opposite population based on the original population:

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

while $X_i(x_{i,1}, x_{i,2}, ..., x_{i,D})$ is the ith individual of the original population and each variable $x_{i,j}$ is bounded by (a_j, b_j).

Several variations of OBL techniques have been proposed, so far [52], [53], [54]. In *quasi-opposition based learning* (QBL), for example, quasi-opposite points are used instead of the actual opposite points [1], [55]. A quasi-opposite point is a randomly generated point located between the opposite point and the middle point (i.e., $a_j + (b_j - a_j)/2$ for $j = 1, ..., D$). More information on other variants of OBL techniques such as quasi-reflection opposition-based learning [56], center-based sampling [57], generalized opposition-based learning [58] and current optimum opposition-based learning [59] is available in [54] and [60].

According to the probability theory, there is 50% chance that an unknown solution is closer to the opposition point than the original point [24]. In [1], Rahnamayan et al. proved that points generated using QBL have more chances to be closer to unknown solutions than points produced by OBL.

OBL and its variants are not the only multi-step techniques that use fitness function as a guideline for enhancing the initial population. Indeed, exploiting objective function to gain some knowledge about fitness landscape is very common in the initialization step [61]. For example, [62] proposed novel local and global selections to generate high-quality initial population for job-shop scheduling. In [63], authors suggest to apply a hill-climbing local search to improve initial population quality. More advanced searches such as quadratic interpolation [12], non-linear local search (a.k.a. simplex) [64], [65], centroid-based sampling [66], Tabu search [67] and smart sampling [68] are also used as the second steps of some compositional initialization techniques.

Although these multi-step techniques achieved good results, they suffer from three main problems; Firstly, these algorithms consume a part of computation budget to evaluate the fitness function and select the best subset of both populations. Secondly, since these techniques calculate the secondary points based on the original population, their performances to some extent depends on the quality of the original points. Based on this fact, some studies proposed to use more advanced point generators for producing the original population [31], [51]. Finally, because of the greediness of the selection mechanism in most of these techniques, the chance of losing informative building blocks at the first stage is very high. In other words, it is very probable that individuals which have useful subcomponents are immediately excluded only due to their low fitness values in comparison with the other individuals'.

Centroidal Voronoi Tessellation (CVT) is a different example of multi-step techniques [69], [70] which does not use fitness function, but other metrics to enhance initial population quality. Generally, CVT tries to partition search space to equal volumes. In the simplest form (a.k.a. Lloyd's algorithm [71]), a temporary population should be generated using PRNG or more sophisticated techniques. Then, by the aid of many randomly generated auxiliary points, search space is divided into some partitions. These partitions and their centres are iteratively enhanced till some criteria are met. Finally, partition centres are used as initial population of EAs [69]. Similar to CVT, simple sequential inhibition process (SSI) is also used in a few studies to produce evenly scattered populations [9].

In comparison with other multi-step techniques, CVT and SSI have two main advantages. Firstly, these algorithms are able to produce geometrically uniform populations without using any objective function evaluations while others evaluate several function evaluations. Secondly, since CVT and SSI do not select pints based on fitness values, it is less likely to miss a great part of search space as greedy selection in some other multi-step techniques sometimes do.

However, CVT also suffers from some problems. Firstly, both CVT and SSI are known as a computationally expensive technique. To lessen this problem, one can hybridize them with QRS or UED techniques (to generate the temporarily population in the first step) in order to increase their convergence speed. Secondly, their performance depends on the internal partitioning (or clustering) algorithm or employed distance measures. Accordingly, practitioners must choose extra parameters (e.g., distance measure and stopping criterion) in addition to EAs' parameters. Thirdly, these iterative techniques might not converge when the population size is relatively small. This situation is more likely to happen when dealing with high dimensional optimization problems.

V. GENERALITY

In this paper, generality of a population initializer refers to the variety of the domains that it can be applied to. In terms of generality, population initialization techniques are grouped into two categories: *generic* and *application specific* techniques.

A. Generic

The population initialization techniques which can be directly applied to all types of optimization problems are called generic techniques. In this sense, all techniques described in previous parts belong to the generic category. These techniques generally assume that the given optimization problem is a black-box puzzle. Hence, no specific knowledge about the region of interest or building blocks of potential solution is

TABLE I. APPLICATION SPECIFIC TECHNIQUES

Authors	Application	Year	Reference
Ma et al.	antenna design	2012	[8]
Dong et al.	circle detection	2012	[28]
Gutierrez	FSS and antenna arrays	2011	[32]
Zhang et al.	flexible job-shop scheduling	2011	[62]
Burke	timetabling	1998	[72]
Garcia et al.	breast cancer prognosis	2007	[73]
Pezzella et al.	flexible job-shop scheduling	2008	[74]
Li at al.	p-median problem	2011	[75]
Tometzki	two-stage stochastic mixed-integer	2011	[76]
Guerrero	segmentation	2012	[77]

available before running the EA. In the absence of such prior knowledge about the problem, generic population initialization techniques can be used easily and effectively [11].

B. Application Specific

The application specific group comprises a few techniques which are specially designed to be applied to particular real-world problems. In the design of such techniques, designers exploit domain knowledge to avoid searching unnecessary regions, producing more promising results and boosting EAs convergence speed. Application specific techniques are potentially beneficial in solving problems that they are specially designed to deal with. However, they may not be effective, efficient or even applicable in many other areas. Consequently, study of these techniques must be only done with experts in those specific domains. Table I presents previously published studies on the application of specific population initialization techniques.

VI. DISCUSSIONS

Although a big and growing body of literature is devoted to the population initialization techniques for EAs, some areas still remain to be explored. The followings are some open research questions which require more investigations.

- Nearly all previous studies, have been done on low dimensional single objective problems (less than 60 dimensions) [11]. Some studies on low dimensions tried to generalize their findings to higher dimensions. However, there has been little agreement on validation of those findings in high dimensional spaces. For example, [69] claimed that the desirable effect of uniformity of initial population is more significant in high dimensions (up to 50 dimensions) while [13], in contrast, claimed that uniform initialization techniques (e.g., QRS) loose their effectiveness in problems of 12 or more dimensions.
- Most comparison studies on population initialization are limited to a few (mostly less than four) techniques. In many cases, the techniques are selected arbitrarily and without considering similarities and dissimilarities of the selected techniques and their categories. This may be due to the lack of a comprehensive categorization. Therefore, the findings cannot easily be generalized to other categories of initialization techniques.
- The relationship between population initialization and other parameters are almost completely neglected in previous works [78]. For example, mutual effect of population size, computational budget, exploration/exploitation ability of the algorithm and the characteristics of the underlying problems are needed to be carefully considered in future comparative studies in order to be able to draw strong conclusions.
- The effect of different population initialization techniques on real-world problems are not explored enough. In other words, most of the existing techniques are only studied on well-known benchmark suites. The potential influence of advanced population initialization techniques on real-world application still needs further investigations.
- Beside some theoretical advices (like this study), too few practical rules of thumb are provided for choosing proper initialization techniques according to different situation. From a practitioners point of view, it is still unclear which initializer matches a specific EA model or suits to a given optimization problem. From this point of view, providing simple rules of thumb for selecting promising initializers in a specific situation is crucial for practitioners.
- Too few studies tried to apply and investigate the potential application of advanced initialization techniques on multi and many objective optimization problems. The effect of these techniques on such problems needs further investigations.

VII. CONCLUSION

This study provided a comprehensive and systematic survey of the existing population initialization techniques for EAs. Three categories were introduced to stamp existing techniques in terms of some exclusive characteristics, i.e., randomness, compositionality and generality. This categorization as well as the representative techniques described in each (sub-)category will benefit EA researchers for choosing from proper state-of-the-art population initialization techniques for their research. The volume of the surveyed techniques revealed that population initialization has become an active research topic in the EA domain. However, many open questions still remain to be resolved. Some of these questions were discussed while more future investigations were advocated.

ACKNOWLEDGEMENT

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

REFERENCES

[1] 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.

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

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] M. Pant, T. Radha, and V. P. Singh, "Particle swarm optimization: Experimenting the distributions of random numbers." in *IICAI*, 2007, pp. 412–420.

[11] 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.

[12] 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.

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

[14] C. Dutang and D. Wuertz, "A note on random number generation," 2009.

[15] L. Smith, *Chaos: a very short introduction*. Oxford University Press, 2007.

[16] S. Ergün and S. Özoguz, "Truly random number generators based on non-autonomous continuous-time chaos," *international journal of circuit theory and applications*, vol. 38, no. 1, pp. 1–24, 2010.

[17] B. Jun and P. Kocher, "The intel random number generator," *Cryptography Research Inc. white paper*, 1999.

[18] J. Soto, "Statistical testing of random number generators," in *Proceedings of the 22nd National Information Systems Security Conference*, vol. 10, no. 99. NIST Gaithersburg, MD, 1999, p. 12.

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

[20] J. Walker, "Hotbits: genuine random numbers," *HotBits: Genuine Random Numbers. September*, 2006.

[21] M. Haahr, "Random. org: True random number service," *School of Computer Science and Statistics, Trinity College, Dublin, Ireland. Website (http://www. random. org). Accessed*, vol. 10, 2010.

[22] P. L'Ecuyer and R. Simard, "Testu01: Ac library for empirical testing of random number generators," *ACM Transactions on Mathematical Software (TOMS)*, vol. 33, no. 4, p. 22, 2007.

[23] M. Pant, R. Thangaraj, and A. Abraham, "Particle swarm optimization: performance tuning and empirical analysis," in *Foundations of Computational Intelligence Volume 3*. Springer, 2009, pp. 101–128.

[24] 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.

[25] F. Panneton, P. L'ecuyer, and M. Matsumoto, "Improved long-period generators based on linear recurrences modulo 2," *ACM Transactions on Mathematical Software (TOMS)*, vol. 32, no. 1, pp. 1–16, 2006.

[26] G. Marsaglia and A. Zaman, "The kiss generator," Tech. rep., Department of Statistics, University of Florida, Tech. Rep., 1993.

[27] 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.

[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] 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.

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

[31] 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*, vol. 17, no. 11, pp. 4316–4327, 2012.

[32] 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.

[33] M. Zhang, W. Zhang, and Y. Sun, "Chaotic co-evolutionary algorithm based on differential evolution and particle swarm optimization," in *Automation and Logistics, 2009. ICAL'09. IEEE International Conference on*. IEEE, 2009, pp. 885–889.

[34] R. Senkerik, D. Davendra, I. Zelinka, and Z. Oplatkova, "Influence of chaotic dynamics on the performance of evolutionary algorithms-an initial study," in *AIP Conference Proceedings*, vol. 1479, 2012, p. 627.

[35] M. Pluhacek, R. Senkerik, I. Zelinka, and D. Davendra, "Chaos pso algorithm driven alternately by two different chaotic maps–an initial study," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 2444–2449.

[36] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons & Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.

[37] R. Senkerik, M. Pluhacek, Z. K. Oplatkova, D. Davendra, and I. Zelinka, "Investigation on the differential evolution driven by selected six chaotic systems in the task of reactor geometry optimization," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 3087–3094.

[38] I. Zelinka, R. Senkerik, and M. Pluhacek, "Do evolutionary algorithms indeed require randomness?" in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 2283–2289.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] K.-T. Fang and D. K. Lin, "Uniform experimental designs and their applications in industry," *Handbook of Statistics*, vol. 22, pp. 131–170, 2003.

[44] 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.

[45] 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.

[46] M. Gong, L. Jiao, F. Liu, and W. Ma, "Immune algorithm with orthogonal design based initialization, cloning, and selection for global optimization," *Knowledge and information systems*, vol. 25, no. 3, pp. 523–549, 2010.

[47] A. B. Owen, *Randomly permuted (t, m, s)-nets and (t, s)-sequences*. Springer, 1995.

[48] J. Dick and F. Pillichshammer, "On the mean square weighted l2 discrepancy of randomized digital (t, m, s)-nets over z2," *Acta Arith*, vol. 117, no. 371-403, pp. 533–560, 2005.

[49] 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.

[50] 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.

[51] 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.

[52] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca, "Enhancing particle swarm optimization using generalized opposition-based learning," *Information Sciences*, vol. 181, no. 20, pp. 4699–4714, 2011.

[53] F. S. Al-Qunaieer, H. R. Tizhoosh, and S. Rahnamayan, "Opposition based computinga survey," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, 2010, pp. 1–7.

[54] M. Ergezer and I. Sikder, "Survey of oppositional algorithms," in *Computer and Information Technology (ICCIT), 2011 14th International Conference on*. IEEE, 2011, pp. 623–628.

[55] 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.

[56] M. Ergezer, D. Simon, and D. Du, "Oppositional biogeography-based optimization," in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*. IEEE, 2009, pp. 1009–1014.

[57] S. Rahnamayan and G. G. Wang, "Center-based sampling for population-based algorithms," in *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE, 2009, pp. 933–938.

[58] 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.

[59] Q. Xu, N. Wang, and R. Fei, "Influence of dimensionality and population size on opposition-based differential evolution using the current optimum," *Information Technology Journal*, vol. 12, pp. 105–112, 2013.

[60] 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.

[61] D. Dasgupta, G. Hernandez, A. Romero, D. Garrett, A. Kaushal, and J. Simien, "On the use of informed initialization and extreme solutions sub-population in multi-objective evolutionary algorithms," in *Computational intelligence in miulti-criteria decision-making, 2009. mcdm'09. ieee symposium on*. IEEE, 2009, pp. 58–65.

[62] G. Zhang, L. Gao, and Y. Shi, "An effective genetic algorithm for the flexible job-shop scheduling problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3563–3573, 2011.

[63] R. Kumar, S. Narula, and R. Kumar, "A population initialization method by memetic algorithm," *International Journal*, vol. 3, no. 4, 2013.

[64] K. Parsopoulos and M. Vrahatis, "Initializing the particle swarm optimizer using the nonlinear simplex method," *Advances in intelligent systems, fuzzy systems, evolutionary computation*, vol. 216, 2002.

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

[66] 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.

[67] M. Sharma and S. Tyagi, "Novel knowledge based selective tabu initialization in genetic algorithm," *International Journal*, vol. 3, no. 5, 2013.

[68] 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.

[69] 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.

[70] Y. Saka, M. Gunzburger, and J. Burkardt, "Latinized, improved lhs, and cvt point sets in hypercubes," *International Journal of Numerical Analysis and Modeling*, vol. 4, no. 3-4, pp. 729–743, 2007.

[71] S. Lloyd, "Least squares quantization in pcm," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.

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

[73] M. García-Arnau, D. Manrique, J. Rios, and A. Rodríguez-Patón, "Initialization method for grammar-guided genetic programming," *Knowledge-Based Systems*, vol. 20, no. 2, pp. 127–133, 2007.

[74] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Computers & Operations Research*, vol. 35, no. 10, pp. 3202–3212, 2008.

[75] X. Li, N. Xiao, C. Claramunt, and H. Lin, "Initialization strategies to enhancing the performance of genetic algorithms for the¡ i¿ p¡/i¿-median problem," *Computers & Industrial Engineering*, vol. 61, no. 4, pp. 1024–1034, 2011.

[76] T. Tometzki and S. Engell, "Systematic initialization techniques for hybrid evolutionary algorithms for solving two-stage stochastic mixed-integer programs," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 2, pp. 196–214, 2011.

[77] J. L. Guerrero, A. Berlanga, and J. M. Molina, "Initialization procedures for multiobjective evolutionary approaches to the segmentation issue," in *Hybrid Artificial Intelligent Systems*. Springer, 2012, pp. 452–463.

[78] B. Kazimipour, X. Li, and A. Qin, "Effects of population initialization in differential evolution for large scale optimization," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014.