J. ADV COMP ENG TECHNOL, 7(1) Winter 2021

An Overview of the Concepts, Classifications, and Methods of Population Initialization in Metaheuristic Algorithms

***Mohammad Reza Hasanzadeh*[1], *Farshid Keynia*[2]**

1- Department of Computer and Information Technology, Islamic Azad University, Kerman Branch, Kerman, IRAN.

2- Department of Energy Management and Optimization, Institute of Science and High Technology and Environmental Sciences, Graduate University of Advanced Technology, Kerman, IRAN. (f.keynia@kgut.ac.ir)

Received (2020-12-16) Accepted (2021-04-22)

Abstract: *In this study, an overview of the concepts, classifications, and different methods of population initialization in metaheuristic algorithms that discussed in recent literatures will be provided. Population initialization is a basic and common step between all metaheuristic algorithms. Therefore, in this study, an attempt has been made that the performance, methods, mechanisms and categories of population initialization as well as the relationship between population initialization and other important parameters in performance and efficiency of metaheuristic algorithms such as search space size, population size, maximum number of iteration, etc., which are mentioned and considered in the literatures, are collected and presented in a regular format.*

Keywords: *Classification, Clustering, Metaheuristic algorithms, Optimization algorithms, Population Initialization.*

How to cite this article:
Mohammad Reza Hasanzadeh, Farshid Keynia. An Overview of the Concepts, Classifications, and Methods of Population Initialization in Metaheuristic Algorithms. J. ADV COMP ENG TECHNOL, 7(1) Winter 2021 : 35-54

Abbreviations:

ABC	Artificial Bee Colony
BCL	Biased Center Learning
CNGs	Chaotic Number Generators
CS	Cuckoo Search
CVT	Centroidal Voronoi Tessellations
DE	Differential Evolution
DRs	Dispatching Rules
EAs	Evolutionary algorithms
ED	Experimental Design
FEs	Function Evaluations
GA	Genetic Algorithm
GLP	Good Lattic Points
GWO	Grey Wolf Optimizer
KNN	K Nearest Neighbor
LD	Low Discrepancy
LHS	Latin Hypercube Sampling
MT	Mersenne Twister
OBL	Opposition-Based Learning
PDF	Probability Density Function
PRNGs	Pseudo-Random Number Generators
PSO	Particle Swarm Optimization
QBL/QOBL	Quasi-Opposition Based Learning
QROBL	Quasi-Reflection Opposition Based Learning
QRS	Quasi-Random Sequence
QRSs	Quasi-Random Sequences
RPL	Ranking Paired Learning
SD	Subgroup Discovery
UED	Uniform Experimental Design
UD	Uniform Design

I. INTRODUCTION

METAHEURISTIC algorithms are typically population-based random search techniques. Fig. 1 illustrates the general framework of a metaheuristic algorithm consisting of its main parts. The sections of a metaheuristic algorithm include setting algorithm parameters, population initialization, global search section, local search section, and checking the stopping conditions in a metaheuristic algorithm. In the parameters setting section, the user can monitor the performance of the metaheuristic algorithm and improve its performance according to the problem under consideration. Some metaheuristic algorithms such as PSO have a number of parameters to adjust. Some algorithms, such as the Whale Optimization Algorithm (WOA), do not have a parameter to adjust. In the initialization section, the algorithm generates an initial population. This initial population is usually randomly generated and improves during iterations of the algorithm until the stop conditions are satisfied. The global search section is responsible for the complete discovery of the search space. This section helps the algorithm to explore global optima points throughout the search space. The local search section is also trying to find the optimal global, among the available solutions. Stop conditions (such as the maximum number of iterations of the algorithm, etc.) are the criteria that help the algorithm to terminate its execution. If any of these criteria are met, the iterations of the algorithm are terminated and the algorithm presents the best solution found so far. In this research, the effect of population initialization methods on the optimization results of metaheuristic algorithms is investigated.

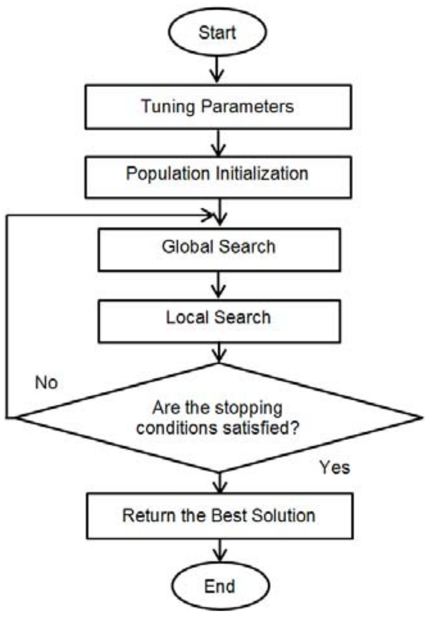

Fig.1. The general framework of a metaheuristic algorithm.

The population initialization is a common step in all metaheuristic algorithms [1], [2]. Experimental observations and numerical simulations have shown that the final solutions of a population-based metaheuristic algorithm depend not only on their specifications but also on the initial starting points for optimization problems, and they can affect the performance of the algorithm to some extent [3]-[7]. One way to improve the production of initial population is to use and employ better quality individuals [8], [9]. The initial population has been considered by researchers with the aim of increasing diversity and exploring more search space. The initial population for metaheuristic optimizers is usually random and based on the principle of guidance to cover the search space as uniformly as possible [2], [9], [10]. However, in some literatures, it has been shown that uniform distributions, especially in the high-dimensional space, are not the best initialization method for solving all optimization problems [10]. In addition, different initialization methods lead to different accuracy for different problems [1]. The purpose of initialization is to provide an initial guess of solutions. Then, during the iterations of the algorithm, these initial solutions are repeatedly improved until the stop conditions are reached. Given that in the black box optimization, there is no prior information about the perspective of a given problem, Therefore, it is expected that a good initial guesses can help the algorithm to find the optimal one, and vice versa, inappropriate guesses prevent the algorithm from finding the optimal one [1], [2], [11].

Given that a good initial population or an inappropriate initial population at the beginning of the algorithm is not predictable, therefore, different methods of initialization in order to cover the appropriate and maximum search space for the given problem have been considered by researchers.

In this regard, different methods of population initialization, such as uniform distribution methods using PRNGs, random distribution methods, clustering methods, partitioning methods and methods based on initial population classification, have been studied in different literatures and the performance of these methods in solving different problems has been evaluated. Among the initialization methods, random methods, uniform distribution, and LHS have been widely used in literatures. The impact of advanced initialization techniques can include increasing the likelihood of finding global optimization, reducing the instability of search results, reducing computational costs, and increasing the quality of the solutions provided by the algorithm [11].

In this paper, an attempt has been made to review the types of initialization methods (includes random, stochastic, deterministic, composite, generic and application specific), types of number generators and initialization techniques (includes PRNGs, number transformations, population-based, CNGs, LHS, Biased, LD, ED, UED, OBL, QBL/QOBL, QRS, clustering and CVT), types of probability distributions in initialization (includes beta, uniform, normal, logarithmic, exponential, Rayleigh, Weibull and random) and types of mappings (includes tent and logistic). Studying the details of these items and the terms used in sections 3 to 6 of the article will help the reader to better understand the article.

The rest of this article includes a history of research in Section 2, classification of initialization methods in Section 3, variants of generators and initialization techniques in Section 4, types of probability distributions in the initialization in Section 5, common mappings used in the initialization in Section 6, study of population initialization methods in interaction with other algorithm parameters and its effect on the performance of metaheuristic in Section 7, selection of the population initialization method in Section 8, the use of existing global search strategies in order to population initialization in Section 9, discussion and analysis in Section 10 and finally the conclusion of the research is presented in Section 11.

II. BACKGROUND

Recently, researchers in the field of optimization have paid special attention to various methods of population initialization and their impact on the performance and efficiency of metaheuristic algorithms [2]. The initialization in metaheuristic optimizers is usually done randomly and based on the principle of uniform coverage of the search space. For example, some recent proposed algorithms, including Supply-demand-based Optimization (SDO) [12], Harris Hawks Optimization (HHO) [13], Most Valuable Player Algorithm [14], Multi-verse Optimize (MVO) [15], and Mayfly Algorithm (MA) [16], use the uniform random initialization method to generate their initial population. Due to the widespread use of metaheuristic algorithms in other scientific fields such as data mining and image processing, attention to the initialization methods in increasing the efficiency and effectiveness of these algorithms in other fields has been considered by a large number of researchers [5], [17]-[20].

In 2020, Li et al. investigated the effect of different initialization methods on the performance of five well-known metaheuristic algorithms in solving 19 test functions. In this study, the effect of 22 different initialization methods using nine probability distributions with different parameters as well as the effect of different population sizes and the number of different iterations on convergence and accuracy of the compared algorithms have been investigated [1].

The probability distributions used include LHS, beta distribution, uniform distribution, normal distribution, logarithmic normal distribution, exponential distribution, Rayleigh distribution, Weibull distribution, random distribution. Research has shown that the use of some initialization methods by some metaheuristic algorithms such as PSO [21] and CS [22], [23], in solving some specific problems, can improve the results of the algorithm, and in contrast, the results of some metaheuristic algorithms, such as DE, are less affected by different initialization methods [24]. The results of this study show that various algorithms can have special performance against different initialization methods in solving different problems, and a strong need for any algorithm is to find the best initialization method to solve a set of given problems [1].

In 2020, Lin et al. proposed a generic method for producing several good solutions in the initial design without additional budget. The proposed method was based on clustering the solution space and feasible solutions were created within the clustered groups with a budget that was dynamically allocated to each group based on the observed information [4].

In 2019, Teng et al. proposed an improved hybrid GWO [25] algorithm. The GWO algorithm is a metaheuristic inspired by the hunting behaviors and social structures of Canis lupus gray wolves. The gray wolf leadership hierarchy includes alpha, beta, delta, and omega wolves, respectively.

Commands are transferred from the highest level of the wolf hierarchy to the lowest level. The hunting mechanism of these wolves also includes three main steps: search for prey (exploration), encircling prey and attacking prey (exploitation). In this algorithm, tent chaotic sequence is used to initialize the position of individuals in order to diversify the group of wolves at the beginning of the algorithm. The results of the research show a better search for optimal global solutions and better algorithm stability than other compared algorithms [26].

In 2019, Sacco and Rios-Coelho used some of the initialization methods to solve three challenging problems, including a nonlinear system, a potential energy function, and a parameter estimation problem. In this research, two different initialization schemes, including MT random generation followed by the application of OBL (as a combined initialization method) and Sobol quasi-random generator [27], were tested against random generation method. Research results have shown that, in terms of effectiveness, the MT initialization method is suitable for small populations and the Sobol sequence method is suitable for large population sizes. In other words, QRSs work better with a large sample size [28].

In 2019, Vlašić et al. examined the performance improvement of the GA [29] using the DRs in the production of the initial population of GA. Research has shown that the use of initialization by DRs, compared to random initialization, leads to significant convergence and better performance, especially when using automatic DRs in GA. In addition, this initialization strategy creates more stable GA results and prevents GA from trapping in the local optimal. The DR-based initialization methods have little effect on GA execution time due to their very low execution time [8].

In 2019, Xue et al. focused on improving PSO performance for feature selection problems in the field of data mining by designing methods for population initialization based on a relief filter [30]. The proposed initialization method involves a combination of mixed initialization and the use of a threshold level. In this method, the KNN is used to evaluate performance in feature selection. Research has shown that the proposed method using KNN classifier to improve PSO performance in solving feature selection problems is promising [17].

In 2018, Gaidhane et al. proposed a hybrid algorithm consisting of GWO and ABC [31] algorithms to improve the performance of complex systems. In this algorithm, a new method has been used to initialize the population based on chaotic mapping and OBL. The purpose of this hybrid initialization method was to produce an initial population with better individuals. Research has shown that the population initialization based on elitism (here, Chaotic mapping mixed with OBL strategy) can provide good diversity (including guess and opposite guess) in candidate solutions for a more appropriate start to optimization [32].

In 2018, Zhang et al. presented a hybrid algorithm based on biogeography-based optimization (BBO) [33] and GWO [25] algorithms. In this algorithm, OBL method is used to prevent the GWO algorithm from trapping in the local optimal [34].

In 2018, de Albuquerque Torreão and Vimieiro studied the performance of EAs in solving the SD task with high-dimensional by biasing the initial population to individuals of smaller sizes. The results show that by reducing the average size of the rules generated for the starting population, interesting results can be achieved even for high-dimensional SD problems when using advanced EAs [5].

In 2018, Łapa et al. presented a population-based initialization algorithm based on negative space. In this algorithm, the negative space is part of the search space defined by individuals who are not satisfactory in terms of fitness function and are determined dynamically during the initialization process. In this algorithm, with the help of negative space, some individuals who are candidates for the population are removed from the initial population without the need for additional evaluation and time consuming. If the candidate is close enough to each of the constituents of the negative space, then an attempt is made to remove it. Based on the results of this study, each benchmark function has a different sensitivity to the different initialization methods used in the algorithm [9].

In 2018, Kazemzadeh Azad, by adding feasible solutions to the initial population (seeding the initial population), examined its effect on the overall performance of metaheuristic techniques. Research has shown that seeding the initial population with feasible solutions can improve the computational efficiency of optimization algorithms, especially in the early stages of optimization [35].

In 2017, Segredo et al. investigated the effect of some initialization methods on the performance of the DE algorithm to solve large-scale optimization problems. Based on the results of this study, it has been confirmed that there was no statistically significant difference between the initialization strategies considered for a significant number of large-scale solved problems. However, in examining the best and worst strategies, there is a significant difference between the initialization mechanisms [36].

In 2017, Łapa et al. proposed a hybrid initialization method. The proposed method is a hybrid of three standard population production methods including number generating methods (such as random method), number transformation methods (transform the generated numbers to cover the search space in a specific way) and methods relied on population (initialization management methods of population). The final population is selected from among the three populations produced above and based on the principle of the fittest in this method. The simulation results confirm the effectiveness of the initialization technique [6].

In 2014, Kazimipour et al. classified different techniques of population initialization into three categories: randomness (includes stochastic and deterministic methods), composite (includes non-composite and composite methods), and generality (include generic and application specific methods). The stochastic methods are divided into two categories: RNGs and CNGs. The deterministic methods are divided into two categories: LD and UED. The composite methods are also divided into hybrid and multi-step methods [11].

In 2014, Kazimipour et al. investigated the effect of population initialization methods on the performance of the DE algorithm in dealing with large-scale optimization problems. Research has shown that initialization plays an important role in improving the performance of the DE algorithm for small population sizes [2].

In 2004, Richards et al. examined the effect of selecting starting points on improving PSO performance with CVT. The results of this study show that although CVT initialization does not improve the performance of the PSO algorithm for low-dimensional problems, it can improve the performance of the algorithm in high-dimensional space [10].

III. CLASSIFICATION OF INITIALIZATION METHODS

Ideally, the optimal solutions found by the algorithm should be independent of the initial selection of their population. Currently, this is only true for special cases such as linear programs and convex optimization. In addition, the vast majority of problems are not linear or convex. Therefore, most algorithms have different levels of dependence on their initial setting [1]. The initial population of a metaheuristic algorithm can be initialized in different ways. In the literatures, random, composite and generality initialization methods have been used. Random initialization is a random sequence of numbers with properties such as complete unpredictability, incompressibility, and irregularity [11]. One type of this technique is stochastic method, which can produce different populations using a different initial seeds. The advantage of stochastic is its ease of use, and the disadvantage of this method is that it does not create a completely uniform population throughout the search space [11], [37]. Another type of random method is deterministic initialization. Deterministic methods focus more on population uniformity using geometric techniques rather than randomness [2], [38]. These techniques, in the absence of prior knowledge of the problem, can improve the capability of the metaheuristic algorithm in the initial iterations with the uniformity of the initial population, resulting in convergence to a better solution and savings of computational budget [2], [11], [28], [36]. In composite Initialization, a number of procedures are used in one technique. The use of composite Initialization in solving complex problems improves the overall diversity of the population [1], [38]. These methods are in the category of greedy methods and do not care about randomness or uniformity of the population. The disadvantage of composite Initialization methods is that they require a higher computational budget than other methods [38]. Generality initialization includes two categories of techniques. The first category is techniques that can be used directly to solve all types of optimization problems. The advantage of this category is the simplicity of implementation, and their disadvantage is that they may not produce appropriate answers to some problems. The second category in this group includes techniques that are specifically designed to solve specific real-world problems. The advantage of this category is that by using the scope of knowledge, they provide the possibility of avoiding the search for unnecessary areas, producing promising results and increasing the speed of convergence. The disadvantage of these methods is that they may not be effective, efficient or practical in many other areas [11], [38]. The details of the above methods are provided below.

1. Random Initialization

A really random sequence is defined as a sequence of numbers with properties such as complete unpredictability, incompressibility, and irregularity. For a more detailed classification, this category can be divided into stochastic and deterministic methods [11].

1) Stochastic Initialization

An initializer is considered to be stochastic if that initializer generates different populations with different initial seeds. Stochastic initialization methods include PRNGs and CNGs [11], [37].

2) Deterministic Initialization

An initializer is considered deterministic if that initializer generates exactly the same population with different initial seeds. Unlike chaotic methods, which use both random and uniform designs to create the initial population, this method only emphasizes uniformity and is usually definitive [36]. Deterministic methods focus more on population uniformity using geometrical techniques rather than randomness [2], [38]. Some research has shown that, despite population size, population uniformity decreases dramatically with increasing population size [39]. This large family of number generators is specifically designed to provide fully distributed points throughout the search space. Using the population initialization with a high level of uniformity can reduce the chance of losing part of the search space in the optimization process. In the absence of prior knowledge of the problem, these techniques can improve the capability of the metaheuristic algorithm in the initial iterations with the uniformity of the initial population, and the consequence is convergence to a better solution and saving computational budget. Deterministic generators are also known as LD (high-uniformity) techniques [2], [11], [28], [36]. Two examples of LD approaches, including QRS and UED, are mentioned in the literatures. Examples of low-discrepancy deterministic initialization methods include Halton [40], Sobol Set (SBL) [27], [28] and GLP [41].

2. Composite Initialization

Methods in which a number of procedures are used in a technique are known as composite methods and are divided into two non-composite and composite categories. Non-composite methods are all techniques that produce populations in just one step. Composite techniques include techniques that produce population in more than one step. Combination techniques can be divided into two categories: hybrid techniques and multi-step techniques. Each component of combination technique can be used separately as a non-composite technique. In solving complex problems, combining two or more distribution methods can improve the overall diversity of the population. These methods may inherit the positive and negative aspects of the basic methods from which they are made [1], [38]. As an example of a hybrid method, a CNG can be used for the initial seed of a PRNG. Multi-step techniques consist of two or more components so that at least one of them cannot be used as an independent initializer. Examples of multi-step methods are the OBL technique and the CVT method. Methods such as OBL and Smart Sampling are also known as two-step initialization methods. Two-step methods are algorithms that create a population in the first step and then improve the population using specific techniques and properties (such as the fitness function) in the second step. Two-step methods are also considered greedy methods and do not care about the randomness or uniformity of the population [38]. As an example of a two-step combination method, we can mention the combination of the MT initialization method with the OBL method [28].

3. Generality Initialization

This group is divided into two categories: generic initialization and application specific initialization.

1) Generic Initialization

Population initialization techniques, which can be used directly for all types of optimization problems, are known as generic techniques. For example, OBL and CVT techniques are known as generic techniques [11].

2) Application Specific Initialization

These methods are specifically designed to solve specific real-world problems. In these methods, designers use the scope of knowledge to avoid the search for unnecessary areas, produce promising results and increase the speed of convergence. The disadvantage of these methods is that they may not be effective, efficient or practical in many other areas [11], [38].

IV. TYPES OF NUMBER GENERATORS AND INITIALIZATION TECHNIQUES

The number generators in initialization are responsible for generating the initial population for a metaheuristic algorithm using their unique technique. Various generators such as PRNGs, methods based on numbers transformations, population-based methods, CNGs, LHS, biased populations, LD, UD, OBL, QBL, QRS, space solution clustering methods and CVT can be used to produce an initial population. The PRNGs generators try to mimic random points using quasi-random numbers. These generators have the advantage of simplicity, the ability to produce a uniform population, the presence in most programming languages and the ability to generate a population without limiting the number of points [28]. These generators are not able to produce fully uniformly distributed points when the search space increases and especially when the population size decreases [11], [18]. The Generators based on number deformation use numbers transformation methods to generate the initial population. These generators have the advantage of ease of use and high speed [9]. The Population-based generators use the population clustering mechanism in initialization. The advantage of this generator is its simplicity in combining with other generators and its disadvantage is the high computational cost [4], [9]. The CNGs generators have a random population generation mechanism. Randomness, unpredictability and regularity are the main properties of these generators [2], [11]. These generators have the advantage of high iteration speeds and can improve the performance of the metaheuristic algorithm in terms of population diversity, success rate, and convergence speed. The disadvantage of the above generators is that they can only be used for small spaces and are very sensitive to initialization conditions, parameter adjustment and implementation accuracy [36]. The LHS generators operate on a space-filling mechanism. The advantage of these generators is that they are easy to implement and prevent excessive accumulation of sample points in one part of the search space. The disadvantage of the above generators is that they do not show a significant advantage for high dimensional problems [1]. The Bias populations have mechanisms that do not use random methods to generate population points. The advantage of these generators is that they can be easily produced from generators such as PRNGs. The LD generators have a mechanism for producing a set of evenly distributed points instead of a random distribution. The advantage of these generators is that they produce more uniform points than PRNGs [38]. The ED generators have mechanisms for generating deterministic uniform points. The disadvantage of these generators is that they are often used to generate discrete numbers. Therefore, some post-processing must be done before using them [38]. The UED generators have a space filling mechanism to create same scattered points in dimensional space [11].

The advantage of these generators is that they can be used for both discrete and continuous dimensional space. The disadvantage of the above generators is that they require a large amount of memory and long computation time in high-dimensional space [38]. The OBL generators operate on asymmetric learning mechanisms. In these generators, instead of considering randomness and/or uniformity, an opposite population is used to select the most suitable individuals [36], [42]. The advantage of these generators is the simplicity of implementation and the presentation of more uniform points. The above generators have disadvantages including high computational cost, dependence of their performance on the quality of the main points of the population and their tendency to lose useful population information due to the greedy of their mechanism [11], [38]. The QBL generators have a mechanism to increase the uniformity of the population. This technique uses quasi-opposite points instead of real opposite points. The advantage of these generators compared to OBL generators is that they produce more uniform points and their disadvantage is having a higher computational overhead [11], [38]. The QRS generators have deterministic mechanisms and do not have any random sequence generation. The advantage of these generators is the simplicity of implementation as well as the production of the same sequences, and their disadvantage is that they do not work well for high-dimensional spaces [38]. The cluster generators operate based on the partitioning of the solution space. The advantage of these generators is the production of more uniform points throughout the dimensional space, especially for high dimensional problems. The disadvantage of these generators is that they do not work well for low dimensional space [4], [10], [42]. The CVT generators have multi-step mechanisms. In these generators, instead of using the fitness function, criteria such as improving the quality of the population are used [10], [11]. The CVT generators have the advantage of generating a uniform geometric population without the use of function evaluation, the ability to access a large part of the search space due to their unique mechanism. Disadvantages of these generators include high computational cost, dependence of their performance on the internal partitioning algorithm, non-convergence when having a small population size [9]. The details of the above generators are provided below.

1. Pseudo-Random Number Generators (PRNGs)

The PRNGs are known as the most common population initializer [2], [38], [42]. These types of generators, due to their simplicity and uniformity, as well as their presence in any programming language, are widely used in the population initialization of metaheuristic algorithms. Due to the lack of limitations in the number of points (population size) and the dimensions size (number of decision variables) in this type of initializer, it makes it possible to use this method for any problem. The PRNG is a deterministic algorithm that generates a sequence $(X_i)_{i\geq0}$ of numbers in the range [0,1) and is merely limited to the period [28]. In cases where the dimensions of the problem are not very high and the population size is large enough, the above generators can create the population initialization with a high level of uniformity. These generators are not able to produce uniformly distributed points when the search space size increases and especially when the population size decreases [11], [18]. The points are designed in a QRS (low- discrepancy) to minimize distance from each other. In addition, Points generated using quasi-random numbers try to mimic random points, while points generated using QRSs try to mimic points with a perfect uniform distribution [28]. An example of an initializer from the PRNGs group is the MT [43], [44] and KISS [45] initialization techniques. We can easily use the "rand" function in MATLAB software to generate PRNGs sequences [46].

2. Number Transformations Methods

This group includes methods for converting pre-generated numbers. In this method, any number of generators can be used as the basis of the method. This group includes space cutting methods with the idea of narrowing the search space, function transformation methods with the idea of concentrating more values in certain spaces of the search space, and normal distribution based methods in which these distributions are used to convert values [9].

3. Population-Based Methods

This group includes all methods that depend on individuals in the initialization. The above methods can be used together with any number generator and any number conversion method. The initialization method based on individual's clustering falls into this category [4], [9]. Data clustering is the process of partitioning a set of data objects into clusters or groups of meaning [47], [48]. The most important factor in the clustering technique is to find the centrality of the clusters and the distance between the samples of each cluster and the center of the cluster [49].

4. Chaotic Number Generators (CNGs)

These generators are used randomly to initialize the population. The above generators have been widely used to improve randomness and uniformity of population initialization [2], [38]. These methods are iterative/recursive algorithms that generate chaotic sequence of uniformly distributed numbers in the [0,1] range. These generators have the advantage of higher iterative speed compared to other methods. An example of a CNG is logistic map [36]. Randomness, unpredictability and regularity are the main properties of the above generators [2], [11]. Proper mapping is required to produce a chaotic population. A one-dimensional chaotic map can be defined as the following equation:

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

where $x_{i,j}^{k+1}$ is the value of jth variable from ith population individual in kth iteration, and μ is the set of parameters defined by the user. Initially, $x_{i,j}^{0}$ is randomly selected and then sequential points are generated using repeated chaotic map [39].

Recent research has shown that the use of chaotic initialization techniques can improve the performance of the metaheuristic algorithm in terms of population diversity, success rate, and convergence speed. Disadvantages of chaotic methods are: They can only be used for low-dimensional space, they are very sensitive to the initial setting conditions and setting their parameters, and they are very sensitive to the accuracy of their implementation. In addition, the presence of some absorbers may cause the population to converge at several fixed points, and it is not yet clear why in some cases a small number of mappings perform significantly better than other mappings. Examples of CNGs-based initialization techniques include tent map, logistics map and sinusoidal map [50] techniques.

5. Latin Hypercube Sampling (LHS)

The LHS method is a spatial filling mechanism. In this method, the search space is considered as a grid and the space is divided into parts with the same distance with the help of the above gird. Then, points are randomly generated within some of these parts. In this method, a set of samples is distributed in such a way that they can be distributed in a limited way in search spaces and effectively prevent the problem of excessive accumulation of sample points in a part of the search space. This method is mainly based on the idea of uniform population expansion in the search space, and in practice provides better expansion than uniform distributions, but does not show a clear advantage for higher dimensions problems. The above method is easy to implement and sometimes works well [1].

6. Biased Populations

Bias means points in the population that are not randomly distributed. Uniform populations generated by PRNGs can easily be converted to biased populations. In this type of population, other distributions such as normal or Gaussian distributions are used.

7. Low-Discrepancy (LD) Methods

Discrepancy means measuring non-uniformity [28], [38], [39]. The LD points are more uniform than those produced by PRNGs. Therefore, The LD methods are mainly used to generate a set of evenly distributed points (instead of random distributions) [38]. An example of this method is Sobol low-discrepancy sequence [27].

8. Experimental Design (ED) Methods

The ED methods are another type of generator that produces deterministic uniform points. Given that The ED methods often generate discrete numbers, some post-processing must be used to solve real-world problems before using them. The UED and OD are examples of this method [38].

9. Uniform Experimental Design (UED) Methods

A UED is a type of space-filling algorithm for the same scattered points in a given space [11]. This technique typically samples a small number of points from a large number of probabilities so that the uniformity of the sample population is maximized. In addition to population uniformity, some of these algorithms can produce sets of points that have additional properties such as orthogonality (production points independent of dimensions). These algorithms are very popular in computer simulation for discrete and continuous dimensional space [39]. The advantage of this method is to accelerate the convergence speed and improve the stability of the metaheuristic algorithm. These methods are generally repetitive and require a large amount of memory and a long calculation time for very high dimensions [38]. Examples of this initializer family include LHS [1], GLP [41], UD [51] and orthogonal design (OD) [45], [52] methods.

10. Opposition-Based Learning (OBL)

The OBL [53], [54] technique is an example of anti-symmetric learning methods that uses an opposing population to select the most appropriate individuals instead of randomly and/or uniformly. In this method, initially a set of points called the main population is produced by methods such as PRNGs, CNGs, UED, etc. Then, by applying simple heuristic rules to the original population, another population of the same size as the opposite population is created. Finally, a subset of the union of both populations is selected based on their fitness values [36], [55]. The following equation produces an opposite population based on the original population:

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

where $X_i = (x_{i,1}, x_{i,2}, \dots, x_{i,D})$ is the vector of ith population individual of the main population and each $x_{i,j}$ variable is in the (a_j, b_j) range.

In anti-symmetric learning methods, the computational cost is doubled due to the doubling of population (as solution nodes for the next generation). Although this method is effective in small dimensions, but its performance is significantly worse for problems with higher dimensions [1], [2], [28], [38]. QBL (QOBL) and QROBL techniques are examples of OBL-based learning techniques [36]. In 2014, Ergezer and Simon reviewed the QROBL method compared to the OBL and QOBL methods, and stated that quasi-reflection opposition individual were probably closer to the optimal solution than opposition individual and quasi-opposition individual [56].

Some of the problems with OBL-based techniques are: high computational budget to evaluate the fitness function and select the best subset of both the main population and the opposite population, dependence of their performance on the quality of the main points in the production of the opposite points, their tendency to lose instructive building blocks (due to their greedy selection mechanisms) [11], [38].

11. Quasi-Opposition-Based Learning (QBL/QOBL) Method

The QBL technique is not a hybrid version of QRS and OBL. In fact, QBL is a modified version of OBL that works to increase population uniformity. In this technique, quasi-opposite points are used instead of real opposite points. A quasi-opposite point is a randomly generated point that is between the opposite point and the middle point $(a_j + \frac{b_j - a_j}{2} \; for \; j = 1,2, \dots, D)$ [11], [38].

12. Quasi-Random Sequence (QRS) Initialization Method

The QRS generators do not have any random sequence production method, and basically these methods have a deterministic nature and no randomness is involved in their algorithm. The QRS method generates the same sequences from a population size and a given dimension size. This method is simple and easy to implement, but it suffers from disaster dimensions [38].

13. Solutions Space Clustering Method

This method is based on the division of solution space. Several clustering methods are mentioned in the literature [4], [10], [55]. For example, in [4] possible solutions are grouped into clusters and a dynamic budget is allocated to each group based on the information observed. This method has been used to produce good solutions in the initial design for optimization problems.

14. Centroidal Voronoi Tessellations (CVT) Methods

The CVT technique is a different example of a multi-step technique. In this method, instead of using the fitness function, other criteria are used to improve the quality of the population. In the CVT method, the search space is divided into the same parts. In its simplest form, first a number of temporary points are generated by a method such as PRNGs, and then with the help of randomly generated points, the search space is divided into several partitions. These partitions and their centers are frequently improved until certain criteria are met. Finally, partition centers are used as the initial population of the metaheuristic algorithm [10], [11]. The two known algorithms for calculating CVT are MacQueen's method [57] and Lloyd's method [58], [59]. The MacQueen's method is probabilistic and requires many repetitions for convergence, but each repetition is relatively simple. In contrast, the Lloyd's method is deterministic and requires only a few iterations, but each repetition is computationally expensive [10].

The advantages of the CVT method are: Produce a uniform geometric population without using function evaluation, a large part of the search space is not lost due to not using greedy selection methods. Disadvantages of this method are: high computational cost, dependence of their performance on the internal partitioning algorithm, non-convergence when having a small population size [11].

V. TYPES OF PROBABILITY DISTRIBUTION IN THE INITIALIZATION

Probability distributions determine how data is distributed. The purpose of measuring data scatter is to find changes and justify them. As the data scatter decreases, the prediction of the value of a random variable becomes more accurate with the help of its mean value. In other words, scatter can indicate the accuracy of a prediction. Dispersion is directly related to how data is distributed. There are many probability distributions such as beta, uniform, normal, logarithmic, exponential, Rayleigh, Weibull and random, some of which can be used for population initialization in metaheuristic algorithms. Different probability distributions have different sampling effects in practice and this effect leads to different degrees of variability in the population of a metaheuristic algorithm. The beta distribution is a continuous probability distribution. Uniform distribution has been used in the initialization of most metaheuristic algorithms. Uniform distribution is easy to implement but is not the best choice for all applications and may even cause premature convergence of algorithm. A normal distribution is a symmetric and continuous distribution. In this distribution, data tends to add up around the mean. Unlike the normal distribution, the logarithmic distribution is an asymmetrical distribution. The exponential distribution is a distribution with an asymmetric long tail. The Rayleigh distribution is one of the statistical distributions with continuous values. The Weibull distribution is a continuous probability distribution. A random distribution is a set of random numbers that follow a certain probability density function [1].

The results of some studies have shown that the use of probability distributions in the initialization of different algorithms has had different effects. For example, for the PSO algorithm, the use of random and Beta distributions in the initialization of the algorithm leads to better results, while for the CS algorithm, beta, exponential and Rayleigh distributions produce better results. Random and uniform methods have been widely used in research on population initialization using probability distributions. In general, the use of some probability distributions such as exponential, beta, and Rayleigh distributions in population initialization results in better algorithm performance [1]. The details of the above probability distributions are provided below.

1. Beta Distribution

Beta distribution is a continuous probability distribution in the (0,1) range. The PDF of this distribution is defined as follows:

$$\mathrm{p(x;a,b)} = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1} \quad (3)$$

where Γ(a) is a standard gamma functions. This distribution has two parameter forms (a<0,b<0) that basically control the distribution form. This distribution is also referred to *asX~Be*(*a*,*b*). The mean value (expected value) is

$\mu = \frac{a}{(a+b)}$ and its variance is $\frac{ab}{[(a+b)^2(a+b+1)]}$.

2. Uniform Distribution

Uniform distribution is widely used in the population initialization in metaheuristic algorithms. The uniform distribution of U [a,b] in the range [a,b] is formulated as follows:

$$\mathrm{p(x)} = \begin{cases} \frac{1}{b-a} & a \le x \le b \\ 0 & otherwise \end{cases} \qquad (4)$$

where the values of *a* and *b* define the range. The average value of this distribution (a + b)/2 and its variance is $\frac{(b-a)^2}{2}$.

3. Normal Distribution (Gaussian Normal Distribution)

Gaussian Normal distribution is one of the most widely used distributions in applications and is not usually used in population initialization. The PDF of this bell distribution can be presented as follows:

$$\mathrm{p(x)} = \frac{1}{\sqrt{2\pi}\delta}\exp\left(-\frac{(x-\mu)^2}{2\delta^2}\right) \qquad (5)$$

where μ and δ indicate the mean and standard deviation, respectively. The display shape of this distribution is $N(\mu,\delta^2)$. The mean, the central position of the probability curve and the standard deviation, determine its expansion on both sides of the mean.

4. Logarithmic Normal Distribution

Unlike normal distribution, Logarithmic distribution is an asymmetrical distribution. The PDF of this distribution is defined as follows:

$$\mathrm{p(x)} = \frac{1}{x\delta\sqrt{2\pi}}\exp\left(-\frac{(lnx-\mu)^2}{2\delta^2}\right) \qquad (6)$$

The random variable *X* that follows this distribution is often written *aslnx~*$N(\mu,\delta^2)$. The mean of this distribution is $e^{[\mu+\frac{\delta^2}{2}]}$ and its variance is $e^{(\delta^2-1)}e^{(2\mu+\delta^2)}$.

5. Exponential Distribution

A long-tail exponential distribution is an asymmetrical distribution. The PDF of this distribution is defined as follows:

$$\mathrm{p(x)} = \begin{cases} \lambda e^{-\lambda x} & x \ge 0 \\ 0 & x < 0 \end{cases} \qquad (7)$$

where λ>0 indicate the distribution parameter and the average distribution is 1/λ and its standard deviation is $1/\lambda^2$.

6. Rayleigh Distribution

The PDF of this distribution is defined as follows:

$$\mathrm{p(x)} = \frac{x}{\delta^2}e^{-\frac{x^2}{2\delta^2}} \qquad \mathrm{x} > 0 \qquad (8)$$

The average of this distribution is $\sqrt{\frac{\pi}{2}}\delta$ and its standard deviation is $(\frac{4-\pi}{2})\delta^2 \approx 0.429\delta^2$.

7. Weibull Distribution

The PDF of this distribution is defined as follows:

$$\mathrm{p(x;\lambda,k)} = \begin{cases} \frac{k}{\lambda}(\frac{x}{\lambda})^{k-1}e^{-(\frac{x}{\lambda})^k} & x \ge 0 \\ 0 & < 0 \end{cases} \qquad (9)$$

where the parameter λ is a scale parameter and κ is a shape parameter. The mean of this distribution is $\lambda\Gamma(1+\frac{1}{\kappa})$ and its variance is

$$\lambda^2[\Gamma\left(1+\frac{2}{k}\right)-\Gamma(1+\frac{1}{k})^2].$$

8. Random Distribution

In general, this distribution has two parameters, mean and variance. The "rand" command in MATLAB software returns a random number between 0 and one. This command generates random numbers with a mean of zero and a variance of unity [46].

VI. MAPS

With the help of mapping, a set of values can be linked to a set of target values. For example, using the f mapping function, the value of x_n can be related to its corresponding x_{n+1} value. One-dimensional mapping is formulated as follows [46]:

$$x_{n+1}=f(x_n) \tag{10}$$

Using the above formula repeatedly will produce a sequence of numbers that are related to each other.

1. Tent Map

Tent mapping has been formulated in the following ways in the literatures [2], [38], [46]:

$$f(x)=\begin{cases} ax & x\le\frac{1}{2} \\ a(1-x) & x>\frac{1}{2}\end{cases} \tag{11}$$

or:

$$f(x)=a\left(\frac{1}{2}-\left|x-\frac{1}{2}\right|\right) \tag{12}$$

Tent mapping can be defined for a member of the population as follows:

$$x_{i,j}^{k+1}=\begin{cases} \mu x_{i,j}^k & for\ x_{i,j}^k<\frac{1}{2} \\ \mu\left(1-x_{i,j}^k\right) & for\ x_{i,j}^k\ge\frac{1}{2}\end{cases} \tag{13}$$

or:

$$x_{i,j}^{k+1}=\mu\left(1-2\left|x_{i,j}^k-0.5\right|\right)\ 0\le x_{i,j}^0\le 1 \tag{14}$$

where $x_{i,j}^k$ is the value of jth variable from ith population individual in kth iteration, and μ value is a positive real constant known as the bifurcation factor. If μ=2 and $x_{i,j}^0\in(0,1)$ value, tent map will produce chaotic sequences.

2. Logistic Map

The formulation of this mapping is as follows:

$$x_{n+1}=rx_n(1-x_n) \tag{15}$$

where x_1 value is the initial condition and r is a parameter. For r<3 value, the solution x_n is a constant mapping. For r=3, the value of x_n oscillates between two values (regardless of the initial value of x_1 value) with a period of 2 according to Fig. 2.

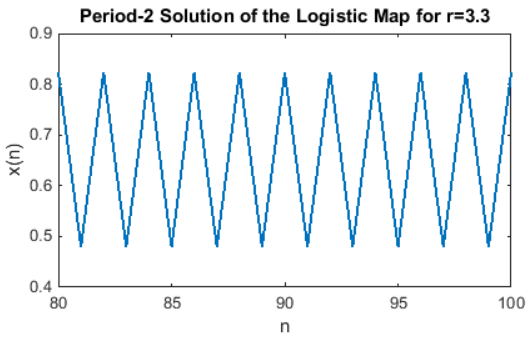

Fig. 2. The logistic map for r=3

As the value of r increases to 3.449, the 4-cycle period of mapping is created, and by exceeding the r value of the critical value of 3.569946, an infinite periodic mapping is created. This will create a chaotic map that is very sensitive to the x_1 initial value. Fig. 3 shows the logistic map for r=3.9 corresponding to the initial conditions x_1=0.1 and x_2=0.100001 values. As can be seen in this figure, in the chaotic state, the logistic map for both initial values is the same at the beginning for n<20 value, but at next times, especially for n>25 value, it is very different [60].

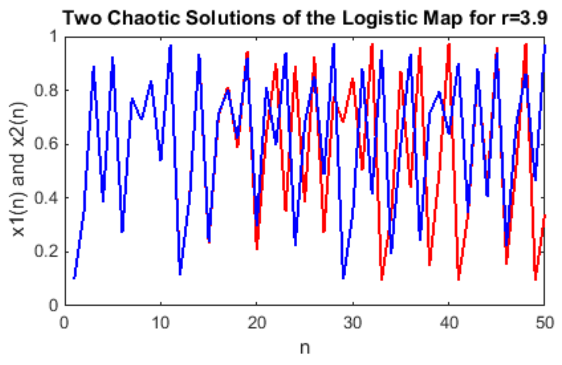

Fig.3. The logistic map in chaotic state with *r* = 3.9 and two different initial values.

VII. POPULATION INITIALIZATION METHODS IN INTERACTION WITH OTHER ALGORITHM PARAMETERS AND ITS EFFECT ON METAHEURISTIC PERFORMANCE

In recent literatures, some researchers have examined the effect of population initialization methods using some probability distributions. In addition, the impact of the main parameters of metaheuristic algorithms, including population size and number of iterations of algorithms, etc., on computational budget, exploration and exploitation capability, and other characteristics of metaheuristic algorithms have also been considered. Due to the fact that the size of the population is always limited, as the size of the search space increases, the chance of the population to cover the promising areas of the search space decreases [10], [11]. For example, the PSO algorithm requires a larger population, and the CS algorithm, with its smaller population size, can produce good results. The DE algorithm can produce better results with more iterations and relatively small population sizes. In contrast, the ABC algorithm is sensitive to the population initialization. At the same time, the population initialization has little effect on the GA algorithm. The accuracy of a metaheuristic algorithm is closely related to two parameters, including population size and the maximum number of repetitions. In short, some algorithms require a large population to achieve the optimal solution, while the rest can find the optimal solution in several iterations with a small population. Some probabilistic distributions, such as exponential, beta and Rayleigh distributions, compared to other distributions, usually lead to better performance of the metaheuristic algorithm [1]. In most population initializer, points are scattered with space-filling designs to explore the entire search domain. Some initializer use exploitation to speed up optimization processes. However, they are application-dependent and require an additional budget [4]. The results of some research challenge the impact of population initialization techniques on large-scale problems. Research has shown that although initialization techniques significantly improve the results of low-dimensional problems, not all of these methods are effective in high-dimensional space. Therefore, a suitable initialization method should be used to solve large-scale problems [2], [36], [38], [39]. The results of some research also indicate that in solving high-scale problems, after selecting the appropriate initialization strategy according to the given problem, tuning the algorithm parameters will have a great impact on the performance and efficiency of the algorithm [36]. Most EAs use PRNGs to initialize their population. Due to the size of the dimensions, this method can negatively affect the behavior of an EA to solve high-dimensional problems. Therefore, researchers have recently noticed the intelligent initialization of candidates. Therefore, researchers have recently considered the intelligent initialization of candidate individuals [18]. In some studies, the relationship between population size and initialization method has also been investigated. The results of these researches show that increasing the size of the population when the computational budget is constant, cannot improve the performance of the generators of basic random numbers. In these cases, some initialization methods can work better than RNGs, regardless of the size of the problem or the size of the population. For example, some methods, such as chaotic numbers, low-discrepancy sequences, and quasi opposition based methods, can improve the performance of metaheuristic algorithms regardless of population size [38]. Some research has also confirmed that there is a lack of uniformity in high dimensions, regardless of the type of EA, the type of initializer or the type of problem. Weak population uniformity is the main reason for the poor performance of advanced initializer in high dimensions. As the dimensions increase linearly, the uniformity of the initial population decreases exponentially. Low uniformity, known as poor coverage or low diversity, dramatically reduces the quality of the initial population [39]. In the absence of useful information about solutions, random initialization techniques are the most common methods of population production. However, in practical applications, sometimes the use of domain knowledge to provide initial population with some promising candidate solutions can improve the efficiency and performance of the metaheuristic algorithm [35]. In solving complex problems, hybridizing two or more distribution methods can improve the overall diversity of the population and, as a result, improve the efficiency and performance of the algorithm [1], [38]. In addition to the advantages of initialization methods, some literatures have pointed to the effect of some common initialization methods on a significant increase in the execution time of algorithms [61].

VIII. SELECTING THE POPULATION INITIALIZATION METHOD

According to studies conducted in the field of population initialization in metaheuristic algorithms, for a specific algorithm, using a population initialization method to solve all problems does not guarantee the best solution by the algorithm. For example, for a multi-start EA algorithm, the randomness of the initial population initialization method must be considered, and to solve a specific problem, the generality of the technique plays an important role in finding the suitable solution to the problem being solved [11]. The choice of population initialization method also depends on the dimensions of the search space. For example, for the PSO algorithm in 2-dimensional space, the randomness of population starting points can provide good coverage of space, even when the starting positions of the particles are not carefully selected. As the size of the search dimension increases, the number of particles in the crowd spreads rapidly according to the total size of the space, and this scattering causes the algorithm to fail [10]. Several studies have also confirmed that the OBL family of techniques improves DE performance on low-dimensional and high-dimensional problems [2]. In high-dimensional problems, the computational budget of an algorithm may also be affected by the population initialization method. Due to the limitations of computational resources, especially in dealing with large-scale problems, the choice of the population initialization method plays an important role in maintaining the efficiency and performance of the metaheuristic algorithm. Some initialization methods use greedy algorithms that use a large population to select the best subset of the initial population. Examples of greedy algorithms are OBL and QBL initializers [2], [36]. Some research has confirmed that some initialization methods are suitable for small population sizes and some are suitable for large population sizes. For example, research has shown that, in terms of effectiveness, the MT initialization method is suitable for small populations and the Sobol sequence method is suitable for large population sizes. In other words, QRSs work better with a large sample size [28].

IX. USING EXISTING GLOBAL SEARCH STRATEGIES TO POPULATION INITIALIZATION IN METAHEURISTIC ALGORITHMS

In addition to articles that explicitly examine and analyze population initialization methods, there are also literatures in the field of metaheuristic algorithms that have specific and unique strategies in their global search. The main purpose of initialization, in addition to creating an initial population for the metaheuristic algorithm, is to have a well-covered population that prevents the algorithm from being trapped in the local optima and explores the entire search space and extracts global optimal points. Due to the fact that in the global search phase of each metaheuristic algorithm, the population is leaded to a more complete coverage than before, and in many metaheuristic algorithms, this phase has a unique strategy. Therefore, according to the functional nature of these strategies and their behavioral similarity with the task of initialization, these strategies can be studied, analyzed and used as population initialization methods and used in initialization tests. The following are examples of these strategies:

In 2020, Segredo et al. used a similarity-based neighborhood search (SNS) method based on the similarity of information obtained from measuring Euclidean distances between solutions in the search space to create a new solution. This method considers the similarity between individuals based on calculating the Euclidean distance. At first, the population is arranged with the most suitable individual in terms of the fitness of its members. Then, part of the list (individuals involved in the search for neighborhood) will be selected according to the given criteria (FEs). This strategy promotes the diversification of the population in the early stages of the optimization process and strengthens the intensification in the final stages of the runs [62]. The above strategy with small FEs can be used as a method for population initialization in a metaheuristic algorithm.

In 2019, Deng et al. presented an improved PSO based on the ranking-based biased learning swarm optimizer. This algorithm has two learning strategies including RPL and BCL. In RPL, worse particles, peer to peer, learn from better particles by their rank. Therefore, the convergence speed will be accelerated. In BCL, each particle learns from the bias center, known as the fitness weighted center, which enhances the algorithm's ability to explore [3]. The BCL strategy can be used as a way to initialize population in other metaheuristic algorithms.

X. DISCUSSION AND ANALYSIS

Table I summarizes the characteristics of some population initialization methods in randomness and composite categories based on the literature review. According to the results of this table, it can be seen that not all listed initialization methods have the ability to solve both low-dimensional and high-dimensional problems. Most initialization methods have a high ability to solve small-scale problems. Among the listed methods, only two initialization methods, including SBL and CVT, have the ability to solve large-scale problems. Where, SBL is not sensitive to population size but CVT requires a large population size.

According to the results of this table, it can be seen that, in order to improve the results for both low-dimensional and high-dimensional problems, the initialization method must be designed and implemented as a combined method (hybrid or multi-step). As in the designed method, a combination of at least two population initial methods, one for high dimensions and the other for low dimensions, which have high capabilities, has been used.

At present, according to recent researches, commenting on which of the low-dimensional methods or which of the high-dimensional methods are suitable for solving a particular problem requires a comprehensive and extensive research. Table II lists some of the research done on population initialization in metaheuristic algorithms.

XI. CONCLUSIONS

In this review article, the concepts, classifications and methods of population initialization in metaheuristic algorithms were studied. The population initialization is one of the main and common steps among all metaheuristic algorithms. The results of many studies on the population initialization in metaheuristic algorithms have shown that the final solutions provided by the algorithms depend on the initial starting points of the population and their quality.

In this research, an attempt was made to study the population initialization in metaheuristic algorithms in terms of general classification of initialization methods, types of number generators, types of probability distribution and types of common maps used in population initialization with subgroups defined for each. In addition, the advantages and disadvantages of different initialization methods and their effects on the performance of metaheuristic algorithms in interaction with dimensional space size (low dimensional problems and large scale problems), population size, type of probability function used, based on the results of recent researches and literatures were presented.

In addition, to reviewing common population initialization methods such as randomness and deterministic methods, the study of some advanced techniques (such as hybrid, filling space, clustering and partitioning techniques) were also considered. Furthermore, an overview of how to select the initialization method in metaheuristic algorithms and also the possibility of using unique techniques and strategies used in the global search section of some algorithms as a method or technique of population initialization of algorithms, especially for subsequent studies were conducted in this field.

As future works in the continuation of this research, the following can be mentioned: Firstly, more focus on population initialization methods in metaheuristic algorithms that are less discussed in the articles, such as GLP, OD, UD, etc. Secondly, a closer look at the strengths and weaknesses of population initialization methods and examining what types of different initialization methods are suitable for solving what problems and why? Thirdly, based on the analysis in section ten of this article, combined methods (hybrid or multi-step) of population initialization should be proposed and studied. So that in these methods, suitable methods for low and high dimensions have been used and the effect of these combined methods on the performance of different metaheuristic algorithms in solving different problems should be investigated.

TABLE I
COMPARISION OF FEATURES OF SOME INITIALIZATION METHODS IN RANDOMNESS AND COMPOSITE CATEGORIES

Category	Subcategory	Type	Method Name	Ability to Solve Problems		The Best Performance in Dimensions	Population Size Required	References
				Low Dimensions	High Dimensions			
Randomness	Stochastic	PRNG	MT	Yes	No	Low	High	[11], [18], [28]
			KISS	Yes	No	Low	High	
		CNG	Tent Map	Yes	No	Low	Insignificant	[38]
			Logistic Map	Yes	No	Low	Insignificant	
			sinusoidal Map	Yes	No	Low	Insignificant	
	Deterministic	LD	SBL	No	Yes	High	Insignificant	[28], [38]
			QRS	Yes	No	Low	High	
			Halton	Yes	No	Low	Insignificant	
		UED	GLP	-	-	-	-	
			OD	-	-	-	-	
			UD	-	-	-	-	
			LHS	Yes	No	Low	-	[1]
Composite	Composite	Multi Step	OBL	Yes	No	Low	Insignificant	[2]
			QOBL/QROBL	Yes	No	Low	Insignificant	
			CVT	No	Yes	High	High	[10]

Note: The dashed items are not mentioned in the literature.

TABLE II
SOME RESEARCH DONE ON POPULATION INITIALIZATION IN METAHEURISTIC ALGORITHMS

Row	Research Title	Author(s)	Year
1	Influence of initialization on the performance of metaheuristic optimizers [1]	Li et al.	2020
2	A Budget Allocation Strategy Minimizing the Sample Set Quantile for Initial Experimental Design [4]	Lin et al.	2020
3	Improving genetic algorithm performance by population initialization with dispatching rules [8]	Vlašić et al.	2019
4	On Initial Populations of Differential Evolution for Practical Optimization Problems [28]	Sacco and Rios-Coelho	2019
5	A Particle Swarm Optimization with Filter-based Population Initialization for Feature Selection [17]	Xue et al.	2019
6	Negative space-based population initialization algorithm (NSPIA) [9]	Łapa et al.	2018
7	Effects of Population Initialization on Evolutionary Techniques for Subgroup Discovery in High Dimensional Datasets [5]	de Albuquerque Torreão and Vimieiro	2018
8	Seeding the initial population with feasible solutions in metaheuristic optimization of steel trusses [35]	Kazemzadeh Azad	2018
9	A Novel Population Initialization Method Based on Support Vector Machine [18]	Keedwell et al.	2018
10	Hybrid initialization in the process of evolutionary learning [6]	Łapa et al.	2017
11	On the comparison of initialization strategies in differential evolution for large scale optimization [36]	Segredo et al.	2017
12	A review of population initialization techniques for evolutionary algorithms [11]	Kazimipour et al.	2014
13	Effects of population initialization on differential evolution for large scale optimization [2]	Kazimipour et al.	2014
14	Choosing a starting configuration for particle swarm optimization [10]	Richards et al.	2004

REFERENCES

1. Q. Li, S.-Y. Liu, and X.-S. Yang, 2020. Influence of initialization on the performance of metaheuristic optimizers, Applied Soft Computing, vol. 91, pp. 106193. https://doi.org/10.1016/j.asoc.2020.106193.

2. B. Kazimipour, X. Li, and A. K. Qin, 2014, July in 2014 IEEE Congress on Evolutionary Computation (CEC) (pp. 2404-2411). IEEE. https://doi.org/10.1109/CEC.2014.6900624.

3. H. Deng, L. Peng, H. Zhang, B. Yang, and Z. Chen, 2019. Ranking-based biased learning swarm optimizer for large-scale optimization, Information Sciences, vol. 493, pp. 120-137. https://doi.org/10.1016/j.ins.2019.04.037.

4. Z. Lin, A. Matta, and S. Du, 2021. A Budget Allocation Strategy Minimizing the Sample Set Quantile for Initial Experimental Design, IISE Transactions, vol. 53, no. 1, pp. 39-57. https://doi.org/10.1080/24725854.2020.1748771.

5. V. de Albuquerque Torreão, and R. Vimieiro, 2018, October in 2018 7th Brazilian Conference on Intelligent Systems (BRACIS) (pp. 25-30). IEEE. https://doi.org/10.1109/BRACIS.2018.00013.

6. K. Łapa, K. Cpałka, and Y. Hayashi, 2017, June in International Conference on Artificial Intelligence and Soft Computing (pp. 380-393). Springer, Cham. https://doi.org/10.1007/978-3-319-59063-9_34.

7. N. Henderson, M. de Sá Rêgo, J. Imbiriba, M. de Sá Rêgo, and W. F. Sacco, 2018. Testing the topographical global initialization strategy in the framework of an unconstrained optimization method, Optimization Letters, vol. 12, no. 4, pp. 727-741. https://doi.org/10.1007/s11590-017-1137-6.

8. Vlašić, M. Đurasević, and D. Jakobović, 2019. Improving genetic algorithm performance by population initialisation with dispatching rules, Computers & Industrial Engineering, vol. 137, pp. 106030. https://doi.org/10.1016/j.cie.2019.106030.

9. K. Łapa, K. Cpałka, A. Przybył, and K. Grzanek, 2018, June in International Conference on Artificial Intelligence and Soft Computing (pp. 449-461). Springer. https://doi.org/10.1007/978-3-319-91253-0_42.

10. M. Richards, and D. Ventura, 2004, July in IEEE Int. Joint. Conf. Neural (pp. 2309-2312). IEEE. https://doi.org/10.1109/IJCNN.2004.1380986.

11. B. Kazimipour, X. Li, and A. K. Qin, 2014, July in 2014 IEEE Congress on Evolutionary Computation (CEC) (pp. 2585-2592). IEEE. https://doi.org/10.1109/CEC.2014.6900618.

12. W. Zhao, L. Wang, and Z. Zhang, 2019. Supply-demand-based Optimization: a Novel Economics-inspired Algorithm for Global Optimization., IEEE Access, vol. 7, pp. 73182-73206. https://doi.org/10.1109/ACCESS.2019.2918753.

13. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, 2019. Harris hawks optimization: Algorithm and applications., Future Generation Computer Systems, vol. 97, pp. 849-872. https://doi.org/10.1016/j.future.2019.02.028.

14. H. Bouchekara, 2020. Most Valuable Player

Algorithm: a novel optimization algorithm inspired from sport., Operational Research, vol. 20, no. 1, pp. 139–195. https://doi.org/10.1007/s12351-017-0320-y.

15. Aljarah, M. Mafarja, A. A. Heidari, H. Faris, and S. Mirjalili, 2020. Multi-verse optimizer: theory, literature review, and application in data clustering, Nature-Inspired Optimizers, pp. 123-141: Springer, Cham. https://doi.org/10.1007/978-3-030-12127-3_8.

16. K. Zervoudakis, and S. Tsafarakis, 2020. A mayfly optimization algorithm, Computers & Industrial Engineering, pp. 106559. https://doi.org/10.1016/j.cie.2020.106559.

17. Y. Xue, W. Jia, and A. X. Liu, 2019, June in 2019 IEEE Congress on Evolutionary Computation (CEC) (pp. 1572-1579). IEEE. https://doi.org/10.1109/CEC.2019.8790156.

18. E. Keedwell, M. Brevilliers, L. Idoumghar, J. Lepagnot, and H. Rakhshani, 2018, October in 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 751-756). IEEE. https://doi.org/10.1109/SMC.2018.00136.

19. A. Yahya, A. Osman, and M. S. El-Bashir, 2017. Rocchio algorithm-based particle initialization mechanism for effective PSO classification of high dimensional data, Swarm and evolutionary computation, vol. 34, pp. 18-32. https://doi.org/10.1016/j.swevo.2016.11.005.

20. N. Dong, C.-H. Wu, W.-H. Ip, Z.-Q. Chen, C.-Y. Chan, and K.-L. Yung, 2012. 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. https://doi.org/10.1016/j.camwa.2012.03.040.

21. R. Poli, J. Kennedy, and T. Blackwell, 2007. Particle swarm optimization., Swarm intelligence, vol. 1, no. 1, pp. 33-57. https://doi.org/10.1007/s11721-007-0002-0.

22. X.-S. Yang, and S. Deb, 2009, December Cuckoo search via Lévy flights., in 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India (pp. 210-214). https://doi.org/10.1109/NABIC.2009.5393690.

23. H. Gandomi, X.-S. Yang, and A. H. Alavi, 2013. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems., Engineering with computers, vol. 29, no. 1, pp. 17-35. https://doi.org/10.1007/s00366-011-0241-y.

24. R. Storn, and K. Price, 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces., Journal of global optimization, vol. 11, no. 4, pp. 341-359. https://doi.org/10.1023/A:1008202821328.

25. S. Mirjalili, S. M. Mirjalili, and A. Lewis, 2014. Grey wolf optimizer., Advances in engineering software, vol. 69, pp. 46-61. https://doi.org/10.1016/j.advengsoft.2013.12.007.

26. Z.-j. Teng, J.-l. Lv, and L.-w. Guo, 2019. An improved hybrid grey wolf optimization algorithm., Soft computing, vol. 23, no. 15, pp. 6617-6631. https://doi.org/10.1007/s00500-018-3310-y.

27. P. Bratley, and B. L. Fox, 1988. Algorithm 659: Implementing Sobol's quasirandom sequence generator, ACM Transactions on Mathematical Software (TOMS), vol. 14, no. 1, pp. 88-100. https://doi.org/10.1145/42288.214372.

28. W. F. Sacco, and A. C. Rios-Coelho, 2019. On Initial Populations of Differential Evolution for Practical Optimization Problems, Computational Intelligence, Optimization and Inverse Problems with Applications in Engineering, pp. 53-62: Springer, Cham. https://doi.org/10.1007/978-3-319-96433-1_3.

29. M. Mitchell, 1995. Genetic algorithms: An overview., Complexity, vol. 1, no. 1, pp. 31-39. https://doi.org/10.1002/cplx.6130010108.

30. Arauzo-Azofra, J. M. Benitez, and J. L. Castro, 2004,December in Proceedings of the fifth international conference on Recent Advances in Soft Computing (pp. 104-109).

31. D. Karaboga, and B. Basturk, 2008. On the performance of artificial bee colony (ABC) algorithm., Applied soft computing, vol. 8, no. 1, pp. 687-697. https://doi.org/10.1016/j.asoc.2007.05.007.

32. P. J. Gaidhane, and M. J. Nigam, 2018. A hybrid grey wolf optimizer and artificial bee colony algorithm for enhancing the performance of complex systems., Journal of computational science, vol. 27, pp. 284-302. https://doi.org/10.1016/j.jocs.2018.06.008.

33. D. Simon, 2008. Biogeography-based optimization., IEEE transactions on evolutionary computation, vol. 12, no. 6, pp. 702-713. https://doi.org/10.1109/TEVC.2008.919004.

34. X. Zhang, Q. Kang, J. Cheng, and X. Wang, 2018. A novel hybrid algorithm based on biogeography-based optimization and grey wolf optimizer., Applied Soft Computing, vol. 67, pp. 197-214. https://doi.org/10.1016/j.asoc.2018.02.049.

35. S. Kazemzadeh Azad, 2018. Seeding the initial population with feasible solutions in metaheuristic optimization of steel trusses, Engineering Optimization, vol. 50, no. 1, pp. 89-105. https://doi.org/10.1080/0305215X.2017.1284833.

36. E. Segredo, B. Paechter, C. Segura, and C. I. González-Vila, 2018. On the comparison of initialisation strategies in differential evolution for large scale optimisation, Optimization Letters, vol. 12, no. 1, pp. 221-234. https://doi.org/10.1007/s11590-017-1107-z.

37. L. Skanderova, and A. Řehoř, 2014. Comparison of pseudorandom numbers generators and chaotic numbers generators used in differential evolution, Nostradamus 2014: Prediction, Modeling and Analysis of Complex Systems, pp. 111-121: Springer, Cham. https://doi.org/10.1007/978-3-319-07401-6_11.

38. B. Kazimipour, X. Li, and A. K. Qin, 2013, June in 2013 IEEE Congress on Evolutionary Computation (pp. 2750-2757). IEEE. https://doi.org/10.1109/CEC.2013.6557902.

39. B. Kazimipour, X. Li, and A. K. Qin, 2014, December in Asia-Pacific Conference on Simulated Evolution and Learning (pp. 479-490). Springer, Cham. https://doi.org/10.1007/978-3-319-13563-2_41.

40. J. H. Halton, 1960. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals, Numerische Mathematik, vol. 2, no. 1, pp. 84-90. https://doi.org/10.1007/BF01386213.

41. H. Sloan, I. Sloan, and S. Joe, 1994, Lattice methods for multiple integration: Oxford University Press.

42. L. Rajashekharan, and C. S. Velayutham, 2016. Is differential evolution sensitive to pseudo random number generator quality?–an investigation, Intelligent Systems Technologies and Applications, pp. 305-313: Springer, Cham. https://doi.org/10.1007/978-3-319-23036-8_26.

43. H. Gandomi, and A. H. Alavi, December 2012. Krill herd: a new bio-inspired optimization algorithm., Communications in nonlinear science and numerical simulation, vol. 17, no. 12, pp. 4831-4845. https://doi.org/10.1016/j.cnsns.2012.05.010.

44. T. Bradley, J. du Toit, R. Tong, M. Giles, and P. Woodhams, 2011. Parallelization techniques for random number generators, GPU Computing Gems Emerald Edition, pp. 231-246: Elsevier. https://doi.org/10.1016/B978-0-12-384988-5.00016-4.

45. Y.-W. Leung, and Y. Wang, 2001. An orthogonal genetic algorithm with quantization for global numerical optimization, IEEE Transactions on Evolutionary computation, vol. 5, no. 1, pp. 41-53. https://doi.org/10.1109/4235.910464.

46. S. Otto, and J. P. Denier, 2005, An introduction to programming and numerical methods in MATLAB.: Springer Science & Business Media. https://doi.org/10.1007/1-84628-133-4.

47. J. R, 2019. A Hybrid Data Clustering Algorithm Using Modified Krill Herd Algorithm and K-MEANS, Journal of Advances in Computer Engineering and Technology, vol. 5, no. 2, pp. 93-106.

48. N. Damya, and F. Soleimanian Gharehchopogh, 2020. An Improved Bat Algorithm based on Whale Optimization Algorithm for Data Clustering, Journal of Advances in Computer Engineering and Technology, pp. -.

49. F. Soleimanian Gharehchopogh, and S. Haggi, 2020. An Optimization K-Modes Clustering Algorithm with Elephant Herding Optimization Algorithm for Crime Clustering, Journal of Advances in Computer Engineering and Technology, vol. 6, no. 2, pp. 79-90.

50. H. Zheng, Y. Zheng, and P. Li, 2013, May in 2013 IEEE International Symposium on Industrial Electronics (pp. 1-5). IEEE. https://doi.org/10.1109/ISIE.2013.6563726.

51. L. Peng, Y. Wang, and G. Dai, 2010, December in 2010 3rd International Symposium on Parallel Architectures, Algorithms and Programming (pp. 239-246). IEEE. https://doi.org/10.1109/PAAP.2010.61.

52. R. Wang, L. Ma, T. Zhang, S. Cheng, and Y. Shi, 2019, June in 2019 IEEE Congress on Evolutionary Computation (CEC) (pp. 262-270). IEEE. https://doi.org/10.1109/CEC.2019.8790307.

53. S. Mahdavi, S. Rahnamayan, and K. Deb, 2018. Opposition based learning: A literature review, Swarm and evolutionary computation, vol. 39, pp. 1-23. https://doi.org/10.1016/j.swevo.2017.09.010.

54. Q. Xu, L. Wang, N. Wang, X. Hei, and L. Zhao, 2014. A review of opposition-based learning from 2005 to 2012, Engineering Applications of Artificial Intelligence, vol. 29, pp. 1-12. https://doi.org/https://doi.org/10.1016/j.engappai.2013.12.004.

55. D. Bajer, G. Martinović, and J. Brest, 2016. A population initialization method for evolutionary algorithms based on clustering and Cauchy deviates, Expert Systems with Applications, vol. 60, pp. 294-310. https://doi.org/10.1016/j.eswa.2016.05.009.

56. M. Ergezer, and D. Simon, 2014. Mathematical and experimental analyses of oppositional algorithms, IEEE transactions on cybernetics, vol. 44, no. 11, pp. 2178-2189. https://doi.org/10.1109/TCYB.2014.2303117.

57. J. MacQueen, 1967, June in Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (pp. 281-297). Oakland, CA, USA.

58. Q. Du, V. Faber, and M. Gunzburger, 1999. Centroidal Voronoi tessellations: Applications and algorithms, SIAM review, vol. 41, no. 4, pp. 637-676. https://doi.org/10.1137/S0036144599352836.

59. S. Lloyd, 1982. Least squares quantization in PCM, IEEE transactions on information theory, vol. 28, no. 2, pp. 129-137. https://doi.org/10.1109/TIT.1982.1056489.

60. F. Barenghi. CHAOS WITH MATLAB, Matlab tutorial, last accessed on January 17 , 2020; http://www.mas.ncl.ac.uk/~ncfb/mat3.pdf.

61. S. Wessing, 2019. Proper initialization is crucial for the Nelder–Mead simplex search, Optimization Letters, vol. 13, no. 4, pp. 847-856. https://doi.org/10.1007/s11590-018-1284-4.

62. E. Segredo, E. Lalla-Ruiz, E. Hart, and S. Voß, 2020. A similarity-based neighbourhood search for enhancing the balance exploration–exploitation of differential evolution, Computers & Operations Research, vol. 117, pp. 104871. https://doi.org/10.1016/j.cor.2019.104871.