Toward Effective Initialization for Large-Scale Search Spaces

Shahryar Rahnamayan
University of Ontario Institute of Technology (UOIT)
Faculty of Engineering and Applied Science
2000 Simcoe Street North
Oshawa, ON, L1H 7K4, Canada
Shahryar.Rahnamayan@uoit.ca

G. Gary Wang
Simon Fraser University (SFU)
Mechatronic Systems Engineering
250-13450 102 Avenue
Surrey, BC, V3T 0A3, Canada
Gary_Wang@sfu.ca

Abstract: Nowadays, optimization problems with a few thousands of variables become more common. Population-based algorithms, such as Differential Evolution (DE), Particle Swarm Optimization (PSO), Genetic Algorithms (GAs), and Evolutionary Strategies (ES) are commonly used approaches to solve complex large-scale problems from science and engineering. These approaches all work with a population of candidate solutions. On the other hand, for high-dimensional problems, no matter what is the individuals' distribution, the population is highly sparse. Therefore, intelligent employment of individual candidates can play a crucial role to find optimal solution(s) faster. The most majority of population-based algorithms utilize pseudo-random population initialization when there is no a priori knowledge about the solution. In this paper, a center-based population initialization is proposed and investigated on seven benchmark functions. The obtained results are compared with the results of Normal, Pseudo Random, and Latin Hypercube population initialization schemes. Furthermore, the advantages of the proposed center-based sampling method are investigated by a mathematical proof and also Monte Carlo (simulation) method. The detailed experimental verifications are provided for problems with 50, 500, and 1000 dimensions.

Key–Words: Population Initialization, Center-Based Sampling, Evolutionary Algorithms, High-Dimensional Search Spaces, Large-Scale Problems.

1 Introduction

Population-based algorithms are utilized to solve real-world complex problems. These algorithms start with a randomly generated candidate solutions when there is no a priori knowledge about the location of the global optima. We call this process population initialization.

There are various sampling methods (such as Normal, Halton, Sobol, and Faure). Applying these methods to initialize the population can affect the best found objective function value. Effects of population initialization are noticeable when we solve real-life problems (mostly expensive optimizations) and when the algorithm has been stopped prematurely because of a long computation time [1]. It means the best found objective function value is different just in early generations. Generally, the effects of population initialization diminish when the dimensionality of the search space increases and the population becomes highly sparse [1]. In the current paper, to address this shortcoming, a new sampling approach, called Center-Based Sampling, for high-dimensional search spaces is proposed. Center-based sampling tries to generate candidate solutions which have a higher chance to be closer to an unknown solution. Given mathematical proofs and reported simulation results in this paper support the proposed sampling method. Furthermore, this method has been utilized to initialize the population for seven benchmark functions (with dimensions of 50, 500, and 1000), then its results have been compared with the results of three other initialization methods. The obtained results for the proposed method are promising.

Sometimes the sampling methods are used not only in the initialization stage, but also during the search, learning, and optimization processes. To mention some examples, Random Search (RS) and Mode-Pursing Sampling (MPS) methods [6, 7] use sampling during the optimization process. The main concern of this paper is that the use of the center-focused populations can help us to solve large-scale problems more efficiently.

The paper is organized as follows: uniform coverage in high-dimensional search spaces is investigated in Section 2. The proposed sampling theory with all corresponding simulation results and a mathematical proof are presented in Section 3. Experimental demonstrations for center-based population initialization are conducted in Section 4. The paper is concluded in Section 5.

2 Uniform Coverage in High-Dimensional Search Spaces

In this section, the varying of population's uniform coverage is investigated on different search space dimensions. Assume the dimension of the problem is D and the selected population size is $N_p = 10 \times D$ (which generally is a large size for any population-based algorithm). For one dimensional space, we have $D = 1$ and $N_p = 10$; suppose we distribute individuals of the population with equal distance (uniformly) over the search interval (e.g., $[a, b]$, Figure 1) and assume that we want to keep the same uniform coverage pattern for higher dimensions as well, see Figure 2 as an example for $2D$ space. In order to have the same uniform coverage for a D-Dimensional space, 10^D individuals are required; whereas, our population size is $10 \times D$ and not 10^D (exponential growth vs. linear increase). By this way, the coverage percent can be calculated by $\frac{10 \times D}{10^D} \times 100$ $(= 10^{3-D} \times D)$; which indicates what percent of the mentioned uniform coverage can be satisfied by the current population size. This coverage percent has been calculated for different dimensions and summarized in Table 1. As seen and not far from our expectation, this value decreases sharply from 100% for $1D$ to less than 0.5% for $4D$. The coverage percent for $D = 50$, $D = 500$, and $D = 1000$ are $5.0000e - 46$, $5.0000e - 495$, and $1.0000e - 994$, respectively, which are very small or close to zero coverages. Nowadays, optimization problems with a few thousands of variables become more prevalent (e.g., structural optimization).

As a consequence, for high-dimensional problems, regardless of population's distribution pattern, achieving a uniform coverage is almost meaningless because the population with a reasonable size is highly sparse to support any distribution pattern. It seems, performance study of the different sampling methods such as Uniform, Normal, Halton, Sobol, Faure, and Low-Discrepancy [1] is valuable only for low-dimensional (non-highly-sparse) populations. In order to tackle with high-dimensional problems efficiently, obviously, we must utilize population's individuals smartly.

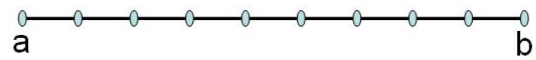

Figure 1: Uniform coverage for $1D$ search space with 10 individuals.

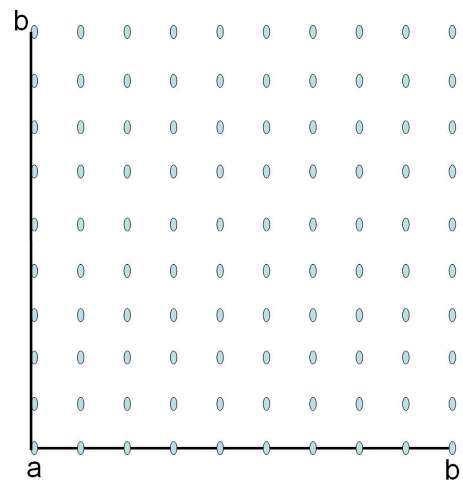

Figure 2: Uniform coverage for $2D$ search space with 10^2 individuals.

3 Center-Based Sampling

Before explaining the proposed sampling theory, we need to conduct some simulations to answer following questions:

1) For a black-box problem (no a priori knowledge about the location of the solution), do all points in the search interval have the same chance to be closer to an unknown solution compared to a randomly generated point?

2) If the answer for the first question is no, what is the pattern for this closeness probability? And whether does this pattern remain the same for all search space dimensions? Following conducted simulation will answer properly to all these questions.

Table 1: D: Dimension, N_p: Population size, N: Required population size for mentioned uniform coverage, Coverage%: Percent of the coverage achieved by given population size, N_p.

D	$N_p = 10 \times D$	$N = 10^D$	Coverage $\% = 10^{3-D} \times D$
1	10	10^1	100
2	20	10^2	20
3	30	10^3	3
4	40	10^4	$4.0000e-01$
5	50	10^5	$5.0000e-02$
6	60	10^6	$6.0000e-03$
7	70	10^7	$7.0000e-04$
8	80	10^8	$8.0000e-05$
9	90	10^9	$9.0000e-06$
10	100	10^{10}	$1.0000e-06$
...	...	...	...
50	500	10^{50}	$5.0000e-46$
...	...	...	...
500	1500	10^{500}	$5.0000e-495$
...	...	...	...
1000	10000	10^{1000}	$1.0000e-994$

3.1 Closeness to Unknown Solution

Let us start with the probability definitions which have been calculated in our simulation.

Definition: The probability of closeness to an unknown solution (s) for the candidate solution (x) and a random point (r) are defined as follows:

$$p_x = p[d(x,s) \leq d(r,s)], \quad (1)$$

$$p_r = p[d(r,s) < d(x,s)], \quad (2)$$

$$p_r + p_x = 1, \quad (3)$$

where d is Euclidean distance function and p stands for probability function.

Algorithm 1 implements our simulation (Monte Carlo method) to calculate p_x, p_r and the average distance of x and r from the solution s, for D-dimensional search space (where x is a D-dimensional vector with the same value for all elements). Figure 3 and Figure 4 depict the results for some sample dimensions (1D, 2D, 3D, 5D, ..., 1000D) graphicly.

As seen in Figure 3, the points which are closer to the center of the search space have a higher chance to be closer to the unknown solution. This chance increases directly with the dimensionality of the search

Algorithm 1 Calculating p_x (probability of closeness of x to a random solution) and $\bar{d}_x$ (average distance of x from a random solution) by the simulation.

$x_i \in [a_i, b_i] = [0,1]$ where $i = 1, 2, 3, ..., D$
TRIALS $\leftarrow 10^6$
for $\vec{x}$ = a to b (stepsize: 10^{-3}, $\vec{x}$ is a vector with the same value for all elements) **do**
  $\bar{d}_x = 0$, $\bar{d}_r = 0$
  $c_r = 0$, $c_x = 0$
  for $R = 1$ to TRIALS **do**
    Generate two random points $\vec{s}$ and $\vec{r}$ in the D-dimensional space (use interval $[0,1]$ for each dimension)
    Calculate the Euclidean distance of $\vec{x}$ and $\vec{r}$ from solution $\vec{s}$ (d_x and d_r)
    $\bar{d}_x \leftarrow \bar{d}_x + d_x$
    $\bar{d}_r \leftarrow \bar{d}_r + d_r$
    if ($d_x \leq d_r$) **then**
      $c_x \leftarrow c_x + 1$
    else
      $c_r \leftarrow c_r + 1$
    end if
  end for
  $\bar{d}_x \leftarrow \bar{d}_x/$TRIALS
  $\bar{d}_r \leftarrow \bar{d}_r/$TRIALS
  $p_x \leftarrow c_x/$TRIALS
  $p_r \leftarrow c_r/$TRIALS
  Save $\bar{d}_x$ and $\bar{d}_r$ for $\vec{x}$
  Save p_x and p_r for $\vec{x}$
end for

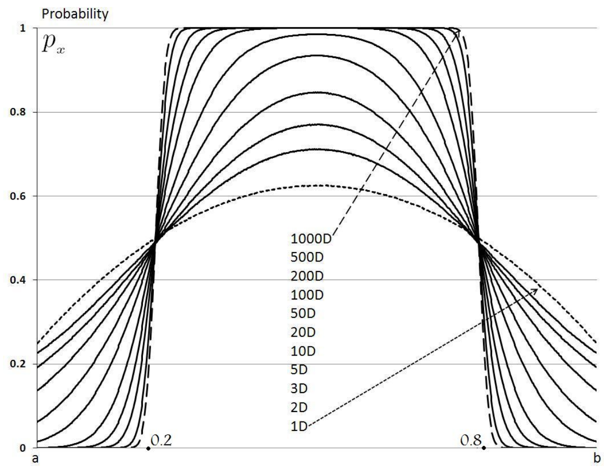

Figure 3: p_x, probability of the closeness of $\vec{x} = [x, x, ..., x]$, $x \in [a, b] = [0, 1]$ (where x is a D-dimensional vector with the same value for all elements) to a uniformly generated random solution compared to the closeness probability of a uniformly generated second random point to that solution. Using a vector with the same value for all elements helps us to show $2D$ map for higher dimensions. By this way, the points on the diameter are investigated.

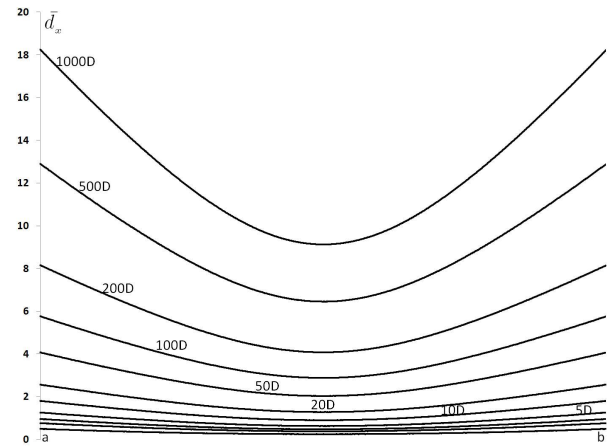

Figure 4: Average distance from random solution ($\bar{d}_x$, $\vec{x} = [x, x, ..., x]$, $x \in [a, b]$) for different search space dimensions.

space. Accordingly, the average distance to the unknown solution is lower for points closer to the center (Figure 4); and similarly such distances decrease sharply for the higher dimensions as the points move closer to the center. Obviously, the center point has the maximum chance to be closer to an unknown solution and at the same time has the minimum average distance from the solution. That is a clear evidence that shows why a center point is a valuable point.

Now, we want to investigate the probability of the closeness of the center point (p_c) to the solution, compared to a second random point. The simulation results are presented in Figure 5. As shown, p_c increases sharply with the dimension and interestingly for the higher dimensions ($D > 30$), it is very close (converges) to one.

Let us look at Figure 3 again, the middle part of the graph is flat when the dimensionality of the search space increases toward a very big number (e.g., 500D or 1000D). It happens in interval $[0.2, 0.8]$ which means 60% of the interval's middle portion. Now, this time we generate a uniform random number in this interval ($U(0.2, 0.8)$, p_c) and compare its closeness to solution with a second uniform random number's closeness generated over the whole interval ($U(0, 1)$, p_r). The result is given in Figure 6. By comparing Figures 5 and 6, we notice that for the first one p_c increases faster than the second one, although, both of them converge to one for higher dimensions ($D > 30$ and $D > 100$ for the first and second graphs, respectively). It was predictable because relaxation of the center point over a sub-interval can reduce the closeness probability value.

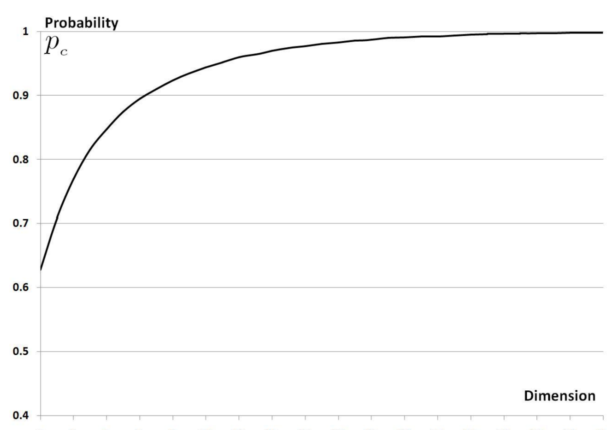

Figure 5: Probability of center-point closeness to solution (compared to a uniformly generated random point, $p_c + p_r = 1$) versus dimension of search space.

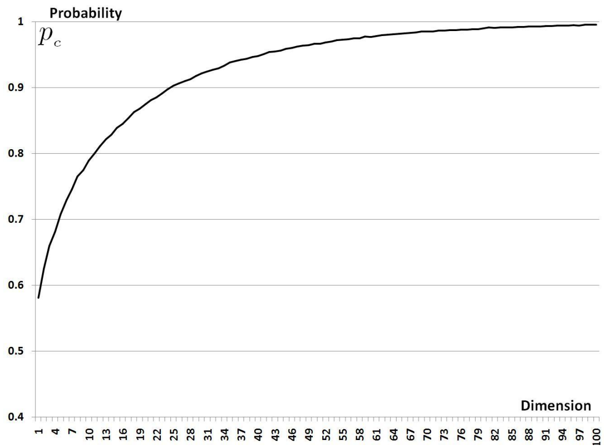

Figure 6: Probability of closeness to solution (a uniform random point generated in $[0.2, 0.8]$ and the second one is generated in $[0, 1]$, $p_c + p_r = 1$) versus dimension of search space.

3.2 Result Analysis

Our simulation results confirm that when the sampling points are closer to the center of the search space they have a higher chance to be closer to an unknown solution. Also on average, their distance from the solution is lower as well. Furthermore, for higher dimensions the mentioned advantages increase sharply; and for very high dimensions (e.g., $D > 1000$) a specific sub-interval (i.e., $[0.2, 0.8]$) presents a flat area for the mentioned probability value ($p \simeq 1$). Also, for these search spaces the population is highly sparse and individuals have a pattern free distribution.

It seems, at least for high-dimensional search spaces, staring with candidate solutions which are biased toward the center of search space, provides a higher chance to be closer to an unknown solution. Converging this probability to $\simeq 1$ when the dimension increases is a strange phenomenon. In the next section, we demonstrate this phenomenon mathematically.

3.3 Mathematical Demonstration

In this section, we will mathematically show that p_c grows with the dimension of the search space and also for higher dimensions it converges to one.

Our calculations are based on the scenario, in which solution is located on border or center of the search spaces (worse case scenarios). This means the solution is far from the center, by this way, we give more chance to other points in the search space to be closer to an unknown solution than the center. Figure 7 presents this situation for a 1D search space. As seen, the solution is located on the boundary and for this case p_c can be calculated as follows:

$$p_{c(D=1)} = 1 - \frac{\frac{a}{2}}{a} = 0.50 \quad (4)$$

Because all points on the illustrated line segment (shadowed region/half of the interval) are closer to the solution than the center point.

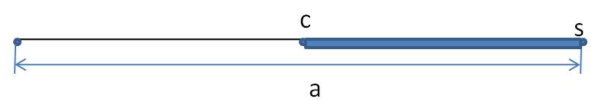

Figure 7: For 1D search space, the solution is located on the boundary. All points on the illustrated line segment (shadowed region, which is $\frac{a}{2}$) are closer to the solution, s, than the center point, c.

For higher dimensions, the calculation procedure is straightforwardly similar. For 2D, shown in Figure 8, all points inside the illustrated circle (shadowed region) are closer to the solution than the center point. The solution on the boundary case for 2D is shown in Figure 9; for this case, p_c can be calculated as follows:

$$p_{c(D=2)} = 1 - \frac{\frac{\pi \times (\frac{a}{2})^2}{2}}{a^2} = 0.61 \quad (5)$$

(i.e., 1-sphere inside 2-cube)

For other dimensions, actually, we should work with hypercubes (i.e., search spaces) and hyperspheres (i.e., sub-spaces where the center is loser for the mentioned scenario). For hypercubes, the edge size is equal to a, and for hyperspheres, the radius is equal to $\frac{a}{2}$. For $D > 2$ dimensions, we can calculate p_c as follows:

$$p_{c(D=3)} = 1 - \frac{\frac{\frac{4}{3}\pi \times (\frac{a}{2})^3}{2}}{a^3} = 0.74 \quad (6)$$

(i.e., 2-sphere inside 3-cube)

$$p_{c(D=4)} = 1 - \frac{\frac{\frac{\pi^2}{2} \times (\frac{a}{2})^4}{2}}{a^4} = 0.85 \quad (7)$$

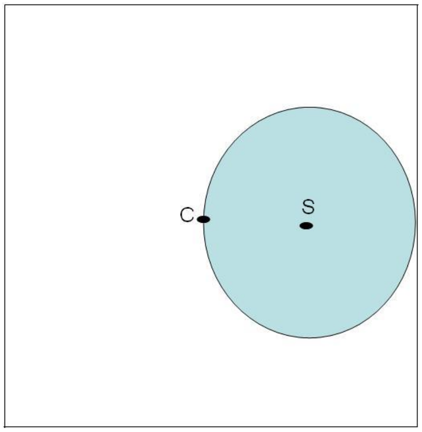

Figure 8: 2D search space. All points inside the illustrated circle (shadowed region) are closer to the solution, s, than the center point, c.

(i.e., 3-sphere inside 4-cube)

$$p_{c(D=5)} = 1 - \frac{\frac{\frac{8\times\pi^2}{15}\times(\frac{a}{2})^5}{2}}{a^5} = 0.92 \quad (8)$$

(i.e., 4-sphere inside 5-cube)

$$p_{c(D=6)} = 1 - \frac{\frac{\frac{\pi^3}{6}\times(\frac{a}{2})^6}{2}}{a^6} = 0.96 \quad (9)$$

(i.e., 5-sphere inside 6-cube)

And finally,

$$p_{c(D=N)} = 1 - \frac{\frac{V_N(\frac{a}{2})}{2}}{a^N} \quad (10)$$

Or

$$p_{c(D=N)} = 1 - \frac{\frac{\frac{\pi^{\frac{N}{2}}\times(\frac{a}{2})^N}{\Gamma(\frac{N}{2}+1)}}{2}}{a^N}, \quad (11)$$

(i.e., (N-1)-sphere inside N-cube)

where $\Gamma(\frac{N}{2}+1)$ is a Gamma Function, for an even N,

$$\Gamma(\frac{N}{2}+1) = (\frac{N}{2})!, \quad (12)$$

and for an odd N,

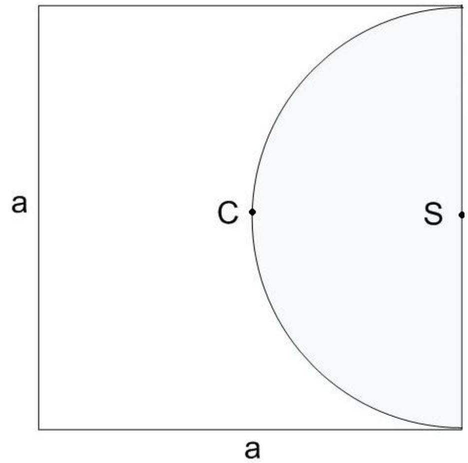

Figure 9: For 2D search space, the solution is located on the border. All points inside the illustrated circle (shadowed region) are closer to the solution, s, than the center point, c.

$$\Gamma(\frac{N}{2}+1) = \sqrt{\pi}\times\frac{N!!}{2^{\frac{(N+1)}{2}}}, \quad (13)$$

where $N!!$ denotes the double factorial.

Hence, for a very big N (very high dimensions), we have:

$$\frac{\frac{V_N(\frac{a}{2})}{2}}{a^N} \approx 0, \quad (14)$$

and so:

$$p_c \approx 1. \quad (15)$$

See Figure 10, for solution-on-corner scenario (this time hyperspheres radius would be $\frac{\sqrt{2}\times a}{2}$ instead of $\frac{a}{2}$, but we have $\frac{1}{4}$ of the hyperspheres instead of $\frac{1}{2}$). Similarly, we have:

$$p_{c(D=N)} = 1 - \frac{\frac{V_N(\frac{\sqrt{2}\times a}{2})}{4}}{a^N} \quad (16)$$

Or

$$p_{c(D=N)} = 1 - \frac{\frac{\frac{\pi^{\frac{N}{2}}\times(\frac{\sqrt{2}\times a}{2})^N}{\Gamma(\frac{N}{2}+1)}}{4}}{a^N}, \quad (17)$$

So for this case, we have:

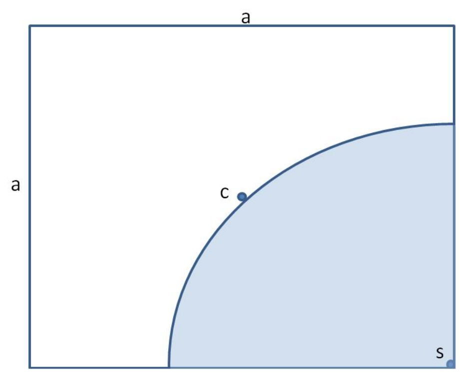

Figure 10: For 2D search space, the solution is located on the corner.

$$p_{c(D=2)} = 0.61 \quad (18)$$

$$p_{c(D=3)} = 0.63 \quad (19)$$

$$p_{c(D=4)} = 0.69 \quad (20)$$

$$p_{c(D=5)} = 0.77 \quad (21)$$

$$p_{c(D=6)} = 0.84 \quad (22)$$

And again for very high dimensions, we have:

$$\frac{\frac{V_N(\frac{\sqrt{2}\times a}{2})}{4}}{a^N} \approx 0, \quad (23)$$

and so:

$$p_c \approx 1. \quad (24)$$

All calculated values for p_c (for different dimensions) by above mentioned approaches are less than the results obtained by the simulation method (Algorithm 1), the reason is that, the current mathematical calculations are based on the worst case scenarios for the center (the solution on the border or corner). Thus, more accurate approximations for the p_c are obtained by the conducted simulation method. The current mathematical demonstrations just support our simulation results.

4 Center-Based Population Initialization

In this section, we want to compare four sampling methods, namely, Center-Based, Pseudo-Random, Latin Hypercube [8], and Normal (or Gaussian) samplings. The center-based sampling is similar to Pseudo-Random sampling but the sampling is performed over a center-focused sub-interval instead of the whole interval. Normal (or Gaussian) sampling is a sort of the center-based sampling because it biases the sampling around the center point. Latin Hypercube Sampling (LHS) ensures that the ensemble of random numbers is representative of the real variability whereas Pseudo-Random sampling is just an ensemble of random numbers without any guarantees. LHS is a highly time consuming method (especially for high-dimensional spaces) compared to three other sampling methods.

By the mentioned sampling methods, we initialize a population (size: $1000 \times D$) for each of seven benchmark functions (with highly different landscapes, for dimensions 50, 500, and 1000). Then, we compare fitness values of the individuals in order to compare sampling methods. In the next section, we briefly review the features of the employed benchmark functions.

4.1 Benchmark Functions

For comparison of different population initialization schemes, a recently proposed benchmark test suite for the CEC-2008 Special Session and Competition on Large Scale Global Optimization [5] has been utilized. It includes two unimodal (F_1-F_2) and five multi-modal (F_3-F_7) functions, among which four of them are non-separable (F_2, F_3, F_5, F_7) and three are separable (F_1, F_4, F_6). Function names and their properties are summarized in Table 2. All benchmark functions are well-known minimization problems. The location of the optimal solution(s) for each function is shifted to a random point(s) in the corresponding search space. By this way, the closeness of the optimal solution(s) to the center or the borders are not known and therefore it supports a fair comparison.

4.2 Experimental Verification

Results of Pseudo-Random, Latin Hypercube, Normal, and Center-Based population initialization for

Table 2: Seven well-known benchmark functions which are utilized for comparison of different population initialization schemes. All of them are scalable and shifted.

Function	Name	Properties	Search Space
F_1	Shifted Sphere Function	Unimodal, Separable	$[-100, 100]^D$
F_2	Shifted Schwefels Problem 2.21	Unimodal, Non-separable	$[-100, 100]^D$
F_3	Shifted Rosenbrocks Function	Multi-modal, Non-separable, A narrow valley from local optimum to global optimum	$[-100, 100]^D$
F_4	Shifted Rastrigins Function	Multi-modal, Separable, Huge number of local optima	$[-5, 5]^D$
F_5	Shifted Griewanks Function	Multi-modal, Non-separable	$[-600, 600]^D$
F_6	Shifted Ackleys Function	Multi-modal, Separable	$[-32, 32]^D$
F_7	FastFractal DoubleDip Function	Multi-modal, Non-separable	$[-1, 1]^D$

seven benchmark functions with the dimensionality of 50 and 500 are reported in Table 3 and Table 4, respectively. The best, worst, mean, median, and Std. of the fitness values for the population individuals are reported in these tables. The population's size is equal to $1000 \times D$. The best result for each case is highlighted in boldface. For the Normal Sampling, the mean and standard deviation are set to 0.5 and 0.15, respectively. The numbers below the word "Center-Based" indicate the center-focused sub-interval's size. For example, the value 0.90 means that the Pseudo-Random sampling is performed just over 90% of the whole interval (the 90% of the interval's center-part in each dimension). By decreasing this value, we are increasing our sampling focus around the center point (generating more center-focused individuals).

4.3 Results Analysis

As seen in Tables 3 and 4, the results for Pseudo-Random and Latin Hypercube samplings are almost the same, although Latin Hypercube sampling is computationally much more expensive. The Normal and Center-Based population initializations compete closely, but for the majority of the functions, Normal Sampling performs slightly better than Center-Based sampling. As mentioned before, both of them focus sampling around the center (but with different intensities in this experiment). For the dimensionality of 1000, Table 5, over all functions (except f_2), Normal initialization performs better than others. For f_2, center-based initialization (0.6) outperforms others.

Let us now increase the sampling intensity around the center for the Center-Based sampling method by changing the sub-interval's size from 0.6 to 0.2. Table 6 reports the results for this experiment ($D = 1000$). As seen, this time, Center-Based sampling outperforms Normal Sampling over all performance metrics (i.e., best, worst, mean, median, and Std.) over all functions. It means when the sampling focuses around the center, then, it generates the individuals with better fitness value.

5 Conclusion Remarks

In this paper, we showed that initial candidates that are closer to the center also have a higher chance to be closer to an unknown optimal solution. Furthermore, this chance increases by the dimension of the search space. This fact was demonstrated via three approaches, namely, the simulation, mathematical reasoning, and population initialization for seven well-known benchmark functions. It is worthwhile to mention that the results of all three approaches confirmed each other. According to presented evidences in this paper, utilizing the center-focused populations to solve large-scale problems is highly promising. It can be expected to be used in many population-based algorithms to increase their acceleration rate, solution accuracy, or robustness, which builds the main directions for our future work. Combining the opposition-based sampling [2, 3, 4] and the proposed center-based sampling is another valuable research area to pursue.

Acknowledgement: Authors would like to thank Dr. K. Tang et al. who shared the Matlab code for benchmark functions.

Table 3: Results for Pseudo-Random, Latin Hypercube, Normal, and proposed Center-Based population initialization on seven benchmark functions with the dimensionality of 50. The best, worst, mean, median, and Std. of the objective values for the population individuals are reported. The population size is equal to $1000 \times D$. The best result for each case is highlighted in **boldface**.

Methods ($D = 50$)		**Random**	**Latin Hypercube**	**Normal**	**Center-Based** 0.95	0.90	0.80	0.70	0.60
f_1	best	235,395	234,956	**171,192**	224,991	217,159	201,007	186,400	177,718
	worse	475,432	475,950	**294,295**	451,828	429,150	388,912	348,440	313,844
	mean	350,610	350,483	**229,053**	334,056	318,903	290,802	265,533	243,834
	median	349,459	349,165	**228,189**	332,934	318,021	289,813	265,004	243,374
	Std	53,451	53,709	**27,139**	50,417	47,087	41,959	36,076	30,360
f_2	best	144.83	144.69	**108.88**	140.96	137.31	130.13	122	116
	worse	194.95	194.99	181.09	190.18	185.13	175.33	165	**155**
	mean	173.33	173.32	138.73	168.86	164.36	155.74	147	**138**
	median	173.86	173.77	**137.33**	169.26	164.69	156	147	138
	Std	11.86	11.81	15.84	11.63	11.38	10.78	10.20	**9.56**
f_3	best	$2.036E+11$	$2.048E+11$	$7.296E+11$	$1.844E+11$	$1.658E+11$	$1.342E+11$	$1.091E+11$	$\mathbf{8.856E+10}$
	worse	$7.055E+11$	$7.066E+11$	$2.688E+11$	$6.326E+11$	$5.673E+11$	$4.481E+11$	$3.520E+11$	$\mathbf{2.735E+10}$
	mean	$4.283E+11$	$4.285E+11$	$1.483E+11$	$3.849E+11$	$3.446E+11$	$2.748E+11$	$2.178E+11$	$\mathbf{1.729E+10}$
	median	$4.222E+11$	$4.221E+11$	$1.434E+11$	$3.794E+11$	$3.394E+11$	$2.715E+11$	$2.150E+11$	$\mathbf{1.710E+10}$
	Std	$1.126E+11$	$1.124E+11$	$4.251E+10$	$1.003E+11$	$9.018E+10$	$7.00E+10$	$5.443E+10$	$\mathbf{4.149E+10}$
f_4	best	1137	1131	**971**	1110	1085	1046	1011	984
	worse	1824	1833	**1372**	1767	1712	1606	1509	1425
	mean	1469	1470	**1165**	1427	1390	1319	1257	1203
	median	1466	1468	**1163**	1425	1388	1318	1256	1202
	Std	153.24	154.91	**89.10**	145.42	138.33	123.96	109.85	97.45
f_5	best	2027	2031	**1440**	1936	1855	1711	1588	1496
	worse	4138	4132	**2507**	3907	3714	3338	2998	2676
	mean	3033	3035	**1938**	2887	2750	2493	2268	2074
	median	3021	3024	**1932**	2878	2743	2487	2263	2070
	Std	468.84	467.78	**236.07**	438.93	411.74	363	312.57	263.72
f_6	best	21.28	21.28	**20.96**	21.25	21.22	21.16	21.08	21.01
	worse	21.81	21.80	**21.61**	21.79	21.78	21.74	21.70	21.65
	mean	21.57	21.57	**21.32**	21.55	21.53	21.49	21.43	21.37
	median	21.58	21.58	**21.33**	21.56	21.54	21.50	21.44	21.38
	Std	0.11	0.11	0.14	0.11	0.12	0.12	0.13	0.13
f_7	best	−415	−415	**−416**	−415	−414	−415	−415	−415
	worse	−255	−255	−255	−256	−255	−256	−255	−255
	mean	−333	−334	−333	−333	−333	−333	−333	−333
	median	−333	−333	−332	−333	−333	−333	−333	−332
	Std	35.65	35.51	35.72	35.36	35.42	35.32	**35.18**	35.38

Table 4: Results for Pseudo-Random, Latin Hypercube, Normal, and proposed Center-Based population initialization on seven benchmark functions with the dimensionality of 500. The best, worst, mean, median, and Std. of the objective values for the population individuals are reported. The population size is equal to $1000 \times D$. The best result for each case is highlighted in **boldface**.

Methods ($D = 500$)		Random	Latin Hypercube	Normal	Center-Based 0.95	0.90	0.80	0.70	0.60
f_1	best	2,930,781	2,931,136	**1,962,733**	2,796,237	2,672,102	2,444,457	2,243,323	2,079,971
	worse	3,948,181	3,946,021	**2,476,166**	3,755,972	3,569,637	3,230,359	2,922,984	2,655,057
	mean	3,429,042	3,429,638	**2,212,550**	3,266,831	3,112,830	2,829,643	2,579,397	2,362,917
	median	3,427,904	3,428,470	**2,211,681**	3,265,726	3,111,654	2,828,928	2,578,819	2,362,505
	Std	167,023	167,131	**84,505**	157,681	148,017	129,868	112,005	95,064
f_2	best	178.03	178.1	**140.23**	173.42	169	160.40	151.72	143.15
	worse	199.65	199.66	219.76	194.67	189.68	179.69	169.70	**159.72**
	mean	192.44	192.45	168.57	187.65	182.84	173.23	163.65	**154.09**
	median	192.92	192.93	167.13	188.09	183.25	173.57	163.95	**154.35**
	Std	3.78	3.78	12.58	3.67	3.57	3.35	3.12	**2.89**
f_3	best	$3.2376E+12$	$3.241E+12$	**$1.101E+12$**	$2.906E+12$	$2.604E+12$	$2.085E+12$	$1.666E+12$	$1.344E+12$
	worse	$5.3847E+12$	$5.385E+12$	**$1.924E+12$**	$4.825E+12$	$4.312E+12$	$3.430E+12$	$2.707E+12$	$2.125E+12$
	mean	$4.2611E+12$	$4.262E+12$	**$1.468E+12$**	$3.821E+12$	$3.419E+12$	$2.727E+12$	$2.164E+12$	$1.713E+12$
	median	$4.2548E+12$	$4.255E+12$	**$1.463E+12$**	$3.815E+12$	$3.414E+12$	$2.723E+12$	$2.161E+12$	$1.711E+12$
	Std	$3.5541E+11$	$3.552E+11$	$1.342E+11$	$3.179E+11$	$2.826E+11$	$2.219E+11$	$1.717E+11$	**$1.306E+11$**
f_4	best	12063	12062	**9561**	11727	11410	10807	10309	9862
	worse	14737	14736	**11144**	14250	13805	12971	12213	11583
	mean	13381	13381	**10340**	12971	12589	11881	11256	10715
	median	13378	13378	**10339**	12969	12588	11880	11255	10714
	Std	441	439	**260.77**	417.24	394.74	356.28	315.24	283.10
f_5	best	24755	24784	**15919**	23548	22395	20322	18505	17000
	worse	33412	33404	**20294**	31681	30055	26995	24289	21868
	mean	29003	28999	**18050**	27538	26148	23602	21349	19402
	median	28992	28989	**18041**	27526	26140	23593	21344	19396
	Std	1429	1426	**719**	1343	1261	1104	952.21	806.18
f_6	best	21.46	21.46	**21.14**	21.44	21.41	21.35	21.28	21.21
	worse	21.68	21.68	**21.42**	21.66	21.64	21.60	21.54	21.48
	mean	21.57	21.57	**21.29**	21.55	21.53	21.46	21.42	21.35
	median	21.58	21.58	**21.29**	21.56	21.53	21.48	21.42	21.35
	Std	0.03	0.03	0.04	0.03	0.03	0.04	0.04	0.04
f_7	best	−3300	−3301	−3300	−3299	−3299	**−3302**	−3297	−3299
	worse	−2688	−2690	−2688	−2686	−2690	−2690	−2689	−2687
	mean	−2992	−2992	−2992	−2991	−2992	−2992	−2990	−2991
	median	−2991	−2992	−2991	−2991	−2991	−2992	−2989	−2991
	Std	100.92	100.71	100.78	100.72	100.58	100.61	**100.48**	100.50

Table 5: Results for Pseudo-Random, Latin Hypercube, Normal, and proposed Center-Based population initialization on seven benchmark functions with the dimensionality of 1000. The best, worst, mean, median, and Std. of the objective values for the population individuals are reported. The population size is equal to $1000 \times D$. The best result for each case is highlighted in **boldface**.

Methods ($D = 1000$)		**Random**	**Latin Hypercube**	**Normal**	**Center-Based** 0.95	0.90	0.80	0.70	0.60
f_1	best	5,995,251	5,989,976	**3,928,389**	5,706,025	5,438,263	4,957,475	4,531,630	4,177,735
	worse	7,495,949	7,504,290	**4,692,386**	7,129,062	6,777,715	6,128,899	5,545,797	5,035,642
	mean	6,735,659	6,735,990	**4,302,486**	6,410,673	6,102,500	5,535,918	5,035,886	4,602,915
	median	6,734,609	6,734,686	**4,301,526**	6,409,455	6,101,372	5,534,756	5,035,234	4,602,572
	Std	233,169	233,009	**117,924**	219,451	206,375	180,939	156,117	132,453
f_2	best	183.65	183.56	148	179	174.46	165.39	156.36	**147.49**
	worse	199.84	199.84	229	194	189.85	179.86	169.86	**159.87**
	mean	194.71	194.71	175	189	184.98	175.28	165.58	**155.92**
	median	195.02	195.02	174	190	185.28	175.56	165.84	**156.14**
	Std	2.77	2.77	11.82	2.70	2.63	2.476	2.31	**2.13**
f_3	best	$2.964E+12$	$6.973E+12$	**2.383E+12**	$6.251E+12$	$5.604E+12$	$4.489E+12$	$3.583E+12$	$2.860E+12$
	worse	$1.023E+13$	$1.023E+13$	**3.626E+12**	$9.172E+12$	$8.200E+12$	$6.532E+12$	$5.164E+12$	$4.062E+12$
	mean	$8.544E+12$	$8.543E+12$	**2.951E+12**	$7.660E+12$	$6.857E+12$	$5.472E+12$	$4.344E+12$	$3.440E+12$
	median	$8.538E+12$	$8.536E+12$	**2.945E+12**	$7.655E+12$	$6.853E+12$	$5.468E+12$	$4.341E+12$	$3.438E+12$
	Std	$5.056E+11$	$5.046E+11$	**1.909E+11**	$4.515E+11$	$4.018E+11$	$3.159E+11$	$2.446E+11$	$1.861E+11$
f_4	best	24658	24655	**19376**	23932	23271	22041	20960	20022
	worse	28683	28678	**21772**	27752	26869	25286	23848	22603
	mean	26643	26642	**20559**	25828	25059	23642	22393	21309
	median	26639	26640	**20558**	25825	25057	23641	22392	21308
	Std	619.27	619.81	**367.21**	587	556	501.52	444.74	397.29
f_5	best	53463	53471	**34875**	50951	48513	44119	40331	37132
	worse	66981	66992	**41707**	63625	60478	54603	49372	44759
	mean	60108	60112	**38210**	57184	54410	49308	44809	40910
	median	60095	60099	**38202**	57173	54403	49300	44803	40905
	Std	2081	2081	**1050**	1960	1844	1613	1392	1180
f_6	best	21.50	21.50	**21.23**	21.48	21.46	21.41	21.35	21.28
	worse	21.66	21.66	**21.43**	21.65	21.63	21.59	21.54	21.48
	mean	21.59	21.59	**21.33**	21.57	21.55	21.50	21.45	21.39
	median	21.59	21.59	**21.33**	21.57	21.55	21.50	21.45	21.39
	Std	0.02	0.02	0.03	0.02	0.02	0.02	0.02	0.03
f_7	best	−6353	−6353	**−6359**	−6351	−6348	−6344	−6344	−6348
	worse	−5438	−5436	**−5445**	−5438	−5440	−5436	−5434	−5437
	mean	−5893	−5893	**−5898**	−5891	−5890	−5888	−5885	−5888
	median	−5892	−5892	**−5897**	−5891	−5889	−5888	−5885	−5888
	Std	140.83	140.78	141.24	140.72	140.47	140.27	**140.00**	140.36

Table 6: Results for Pseudo-Random, Latin Hypercube, Normal, and proposed Center-Based population initialization on seven benchmark functions with the dimensionality of 1000. The best, worst, mean, median, and Std. of the objective values for the population individuals are reported. The population size is equal to $1000 \times D$. The best result for each case is highlighted in **boldface**.

Methods		Normal	Center-Based (0.20)
f_1	best	$3,928,389$	$\mathbf{3,398,099}$
	worse	$4,692,386$	$\mathbf{3,675,275}$
	mean	$4,302,486$	$\mathbf{3,536,123}$
	median	$4,301,526$	$\mathbf{3,536,040}$
	Std	$117,924$	$\mathbf{42,724}$
f_2	best	148	$\mathbf{112.92}$
	worse	229	$\mathbf{119.92}$
	mean	175	$\mathbf{117.67}$
	median	174	$\mathbf{117.80}$
	Std	11.82	$\mathbf{1.23}$
f_3	best	$2.383E+12$	$\mathbf{1.366E+12}$
	worse	$3.626E+12$	$\mathbf{1.642E+12}$
	mean	$2.951E+12$	$\mathbf{1.502E+12}$
	median	$2.945E+12$	$\mathbf{1.501E+12}$
	Std	$1.909E+11$	$\mathbf{4.267E+10}$
f_4	best	19376	$\mathbf{17834}$
	worse	21772	$\mathbf{19451}$
	mean	20559	$\mathbf{18642}$
	median	20558	$\mathbf{18642}$
	Std	367.21	$\mathbf{249.47}$
f_5	best	34875	$\mathbf{30075}$
	worse	41707	$\mathbf{32544}$
	mean	38210	$\mathbf{31310}$
	median	38202	$\mathbf{31309}$
	Std	1050	$\mathbf{381}$
f_6	best	21.23	$\mathbf{21.05}$
	worse	21.43	$\mathbf{21.22}$
	mean	21.33	$\mathbf{21.14}$
	median	21.33	$\mathbf{21.14}$
	Std	0.03	$\mathbf{0.02}$
f_7	best	-6359	$\mathbf{-6367}$
	worse	-5445	$\mathbf{-5488}$
	mean	-5898	$\mathbf{-5925}$
	median	-5897	$\mathbf{-5925}$
	Std	141.24	$\mathbf{135.70}$

References:

[1] H. Maaranen, K. Miettinen, A. Penttinen, *On initial populations of a genetic algorithm for continuous optimization problems*, Journal of Global Optimization, Vol. 37, No. 3, 2007, pp. 405-436.

[2] S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, *Opposition versus Randomness in Soft Computing Techniques*, Elsevier Journal on Applied Soft Computing, Vol. 8, March 2008, pp. 906-918.

[3] S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, *Opposition-Based Differential Evolution*, IEEE Transactions on Evolutionary Computation, Vol. 12, Issue 1, Feb. 2008, pp. 64-79.

[4] S. Rahnamayan, G. Gary Wang, *Solving Large Scale Optimization Problems by Opposition-Based Differential Evolution (ODE)*, World Scientific and Engineering Academy and Society, Transactions on Computers, Vol. 7, Issue 10, Oct. 2008, pp. 1792-1804.

[5] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, Z. Yang, *Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization*, Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, http://nical.ustc.edu.cn/cec08ss.php, 2007.

[6] B. Sharif, G.G. Wang, and T. El-Mekkawy, *Mode Pursing Sampling Method for Discrete Variable Optimization on Expensive Black-box Functions*, ASME Transactions, Journal of Mechanical Design, Vol. 130, 2008, pp.021402-1-11.

[7] L. Wang, S. Shan, and G.G. Wang, *Mode-Pursuing Sampling Method for Global Optimization on Expensive Black-box Functions*, Journal of Engineering Optimization, Vol. 36, No. 4, August 2004, pp. 419-438.

[8] M. D. McKay, R. J. Beckman and W. J. Conover, *A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from A Computer Code*, Technometrics, Vol. 21, No. 2, May 1979, pp. 239-245.

[9] S. Rahnamayan, G. Gary Wang, *Solving Large Scale Optimization Problems by Opposition-Based Differential Evolution (ODE), World Scientific and Engineering Academy and Society, Transactions on Computers*, Vol. 7, Issue 10, Oct. 2008, pp. 1792-1804.

[10] S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, *Learning Robust Object Segmentation from User-Prepared Samples, WSEAS Transactions on Computers*, Vol. 4, Issue 9, Sep. 2005, pp. 1163-1170.

[11] S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, *Towards Incomplete Object Recognition, World Scientific and Engineering Academy and Society, Transactions on Systems*, Vol. 4, Issue 10, October 2005, pp. 1725-1732.