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.

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, $N_p$.

<table>
<thead>
<tr>
<th>$D$</th>
<th>$N_p = 10 \times D$</th>
<th>$N = 10^D$</th>
<th>Coverage % = $10^D - D \times N_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10, $10^3$</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20, $10^4$</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>30, $10^5$</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>40, $10^6$</td>
<td>4,000,000e-01</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>50, $10^7$</td>
<td>5,000,000e-02</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>60, $10^8$</td>
<td>6,000,000e-03</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>70, $10^9$</td>
<td>7,000,000e-04</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>80, $10^{10}$</td>
<td>8,000,000e-05</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>90, $10^{11}$</td>
<td>9,000,000e-06</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>100, $10^{12}$</td>
<td>1,000,000e-06</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>500, $10^{50}$</td>
<td>5,000,000e-46</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>1500, $10^{500}$</td>
<td>5,000,000e-495</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>100000, $10^{1000}$</td>
<td>1,000,000e-994</td>
<td></td>
</tr>
</tbody>
</table>

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) graphically.

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 space.
Figure 3: $p_c$, 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.
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:

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

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

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:

\[
p_{c(D=2)} = 1 - \frac{\pi \times (\frac{a}{2})^2}{a^2} = 0.61
\] (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:

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

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

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

Figure 6: Probability of closeness to solution (a uniform random point generated in \([0, 0.8]\) and the second one is generated in \([0, 1]\), \(p_c + p_r = 1\)) versus dimension of search space.
(i.e., 3-sphere inside 4-cube)
\[ p_{c(D=5)} = 1 - \frac{\frac{8 \times \pi^2 \times (\frac{a}{2})^5}{15}}{a^5} = 0.92 \] (8)

(i.e., 4-sphere inside 5-cube)
\[ p_{c(D=6)} = 1 - \frac{\frac{\pi^3 \times (\frac{a}{2})^6}{2}}{a^6} = 0.96 \] (9)

(i.e., 5-sphere inside 6-cube)
And finally,
\[ p_{c(D=N)} = 1 - \frac{V_N(\frac{a}{2})}{\frac{a^N}{2}} \] (10)

Or
\[ p_{c(D=N)} = 1 - \frac{\pi \times (\frac{\sqrt{2} \times a}{2})^N}{\Gamma(\frac{N+1}{2})} \] (11)

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

where \( \Gamma(\frac{N}{2} + 1) \) is a Gamma Function, for an even \( N \),
\[ \Gamma(\frac{N}{2} + 1) = (\frac{N}{2})! \] (12)

and for an odd \( N \),
\[ \Gamma(\frac{N}{2} + 1) = (\frac{N}{2}) \Gamma(\frac{N}{2}) \] (13)

where \( N!! \) denotes the double factorial.

Hence, for a very big \( N \) (very high dimensions), we have:
\[ \frac{V_N(\frac{a}{2})}{\frac{a^N}{2}} \approx 0, \] (14)
and so:
\[ p_c \approx 1. \] (15)

See Figure 10, for solution-on-corner scenario (this time hyperspheres radius would be \( \sqrt{\frac{\pi}{2}} \times a \) instead of \( \frac{\sqrt{\pi}}{2} \), but we have \( \frac{N}{2} \) of the hyperspheres instead of \( \frac{1}{2} \)). Similarly, we have:
\[ p_{c(D=N)} = 1 - \frac{V_N(\frac{\sqrt{\pi} \times a}{2})}{\frac{a^N}{4}} \] (16)

Or
\[ p_{c(D=N)} = 1 - \frac{\pi \times (\frac{\sqrt{2} \times a}{2})^N}{\Gamma(\frac{N+1}{2})} \] (17)

So for this case, we have:
For 2D search space, the solution is located on the corner.

\[ p_{c(D=2)} = 0.61 \] (18)

\[ p_{c(D=3)} = 0.63 \] (19)

\[ p_{c(D=4)} = 0.69 \] (20)

\[ p_{c(D=5)} = 0.77 \] (21)

\[ p_{c(D=6)} = 0.84 \] (22)

And again for very high dimensions, we have:

\[ \frac{V_N(\sqrt{2} \times a^2)}{a^N} \approx 0, \] (23)

and so:

\[ p_c \approx 1. \] (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.

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

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.
population size is equal to 1000 × D. The best result for each case is highlighted in boldface.

The best result for each case is highlighted in boldface.
<table>
<thead>
<tr>
<th></th>
<th>100</th>
<th>50</th>
<th>48</th>
<th>2991</th>
<th>2991</th>
<th>2992</th>
<th>2992</th>
<th>2688</th>
<th>2689</th>
<th>2690</th>
<th>2690</th>
<th>27538</th>
<th>27538</th>
<th>29003</th>
<th>29003</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 01</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>39 39</td>
<td>40 40</td>
<td>40 40</td>
<td>40 40</td>
<td>40 40</td>
<td>40 40</td>
<td>40 40</td>
<td>40 40</td>
<td>40 40</td>
<td>40 40</td>
<td>40 40</td>
<td>40 40</td>
<td>40 40</td>
<td>40 40</td>
<td>40 40</td>
<td></td>
</tr>
<tr>
<td>12 12</td>
<td>21 21</td>
<td>21 21</td>
<td>21 21</td>
<td>21 21</td>
<td>21 21</td>
<td>21 21</td>
<td>21 21</td>
<td>21 21</td>
<td>21 21</td>
<td>21 21</td>
<td>21 21</td>
<td>21 21</td>
<td>21 21</td>
<td>21 21</td>
<td></td>
</tr>
<tr>
<td>98 98</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td></td>
</tr>
<tr>
<td>98 98</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td></td>
</tr>
<tr>
<td>98 98</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td>29 29</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Results for pseudo-random Latin Hypercube Normal and proposed center-based population initialization on seven benchmark functions with the improvement of 500. The best result for each case is highlighted in boldface.

The best result is equal to 1000 \( \times D \). The best result for each case is highlighted in boldface.
Table 5: Results for Pseudo-Random, Latin Hypercube, Normal, and proposed Center-Based population initialization on seven benchmark functions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Pseudo-Random</th>
<th>Latin Hypercube</th>
<th>Normal</th>
<th>Center-Based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Griewangk</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Ackley</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Griewangk</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Ackley</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

The best result for each case is highlighted in boldface.
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.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Normal</th>
<th>Center-Based (0.20)</th>
</tr>
</thead>
<tbody>
<tr>
<td>best</td>
<td>3,928,389</td>
<td>3,398,099</td>
</tr>
<tr>
<td>worse</td>
<td>4,692,386</td>
<td>3,675,275</td>
</tr>
<tr>
<td>$f_1$</td>
<td>mean 4,302,486</td>
<td>3,536,123</td>
</tr>
<tr>
<td>median</td>
<td>4,301,526</td>
<td>3,536,040</td>
</tr>
<tr>
<td>Std</td>
<td>117,924</td>
<td>42,724</td>
</tr>
<tr>
<td>best</td>
<td>148</td>
<td>112.92</td>
</tr>
<tr>
<td>worse</td>
<td>229</td>
<td>119.92</td>
</tr>
<tr>
<td>$f_2$</td>
<td>mean 175</td>
<td>117.67</td>
</tr>
<tr>
<td>median</td>
<td>174</td>
<td>117.80</td>
</tr>
<tr>
<td>Std</td>
<td>11.82</td>
<td>1.23</td>
</tr>
<tr>
<td>best</td>
<td>$2.383E+12$</td>
<td>$1.366E+12$</td>
</tr>
<tr>
<td>worse</td>
<td>$3.626E+12$</td>
<td>$1.642E+12$</td>
</tr>
<tr>
<td>$f_3$</td>
<td>mean $2.951E+12$</td>
<td>$1.502E+12$</td>
</tr>
<tr>
<td>median</td>
<td>$2.945E+12$</td>
<td>$1.501E+12$</td>
</tr>
<tr>
<td>Std</td>
<td>$1.909E+11$</td>
<td>$4.267E+10$</td>
</tr>
<tr>
<td>best</td>
<td>19376</td>
<td>17834</td>
</tr>
<tr>
<td>worse</td>
<td>21772</td>
<td>19451</td>
</tr>
<tr>
<td>$f_4$</td>
<td>mean 20559</td>
<td>18642</td>
</tr>
<tr>
<td>median</td>
<td>20558</td>
<td>18642</td>
</tr>
<tr>
<td>Std</td>
<td>367.21</td>
<td>249.47</td>
</tr>
<tr>
<td>best</td>
<td>34975</td>
<td>30075</td>
</tr>
<tr>
<td>worse</td>
<td>41707</td>
<td>32544</td>
</tr>
<tr>
<td>$f_5$</td>
<td>mean 38210</td>
<td>31310</td>
</tr>
<tr>
<td>median</td>
<td>38202</td>
<td>31309</td>
</tr>
<tr>
<td>Std</td>
<td>1050</td>
<td>381</td>
</tr>
<tr>
<td>best</td>
<td>21.23</td>
<td>21.05</td>
</tr>
<tr>
<td>worse</td>
<td>21.43</td>
<td>21.22</td>
</tr>
<tr>
<td>$f_6$</td>
<td>mean 21.33</td>
<td>21.14</td>
</tr>
<tr>
<td>median</td>
<td>21.33</td>
<td>21.14</td>
</tr>
<tr>
<td>Std</td>
<td>0.03</td>
<td>0.02</td>
</tr>
<tr>
<td>best</td>
<td>$-6359$</td>
<td>$-6367$</td>
</tr>
<tr>
<td>worse</td>
<td>$-5445$</td>
<td>$-5488$</td>
</tr>
<tr>
<td>$f_7$</td>
<td>mean $-5898$</td>
<td>$-5925$</td>
</tr>
<tr>
<td>median</td>
<td>$-5897$</td>
<td>$-5925$</td>
</tr>
<tr>
<td>Std</td>
<td>141.24</td>
<td>135.70</td>
</tr>
</tbody>
</table>

References:


