Evaluation of Initialization Methods for the Archerfish Hunting Optimizer: A Comparative Study

Aridj Ferhat[1,*,†], Farouq Zitouni[1,†], Abdelhadiimane Limane[1,†], Rihab Lakbichi[1,†] and Saad Harous[2,†]

[1]*Department of Computer Science and Information Technology, Laboratory of Artificial Intelligence and Information Technology, Kasdi Merbah University – Ouargla, Algeria*

[2]*College of Computing and Informatics, Department of Computer Science, University of Sharjah – Sharjah, UAE*

Abstract

This study investigates the impact of different initialization methods on the Archerfish Hunting Optimizer (AHO) performance. We first examined how variations in population size and the maximum number of iterations affect the algorithm's performance. Subsequently, we evaluated six probability distributions and three low-discrepancy sequences for initializing the AHO's population. The performance of these initialization methods was compared to the original AHO, which uses uniform distribution for initialization. Statistical analyses were conducted using the Friedman and Wilcoxon signed-rank tests for post-hoc comparisons. The findings offer insights into how different initialization methods influence AHO's effectiveness, providing guidance for selecting optimal initialization strategies to improve its performance.

Keywords

Metaheuristic Algorithms, Population Initialisation, Probability Distributions, Low-Discrepancy Sequences, Archerfish Hunting Optimizer.

1. Introduction

Optimization is a fundamental concept in various fields such as artificial intelligence [1], computer science [2], machine learning [3], and operations research [4]. It involves finding the best possible solution, either a minimum or maximum, to a given objective function. Traditional optimization techniques, including linear programming [5], quadratic programming [6], convex optimization [7], and interior-point methods [8], have been widely used to address these challenges. However, the growing complexity and scale of real-world problems have led researchers to explore more flexible and robust approaches, such as Metaheuristic Algorithms (MAs), which have gained significant attention in recent years.

MAs are high-level problem-independent techniques that explore the search space to find near-optimal solutions within a reasonable computational time. They are inspired by natural processes and phenomena, including biological evolution, animal behaviours, and physical principles. Examples of widely used MAs include the genetic algorithm [9], inspired by the process of natural selection, the particle swarm optimization [10], which mimics the social behaviour of bird flocking and fish schooling, the ant colony optimization [11], based on the foraging behaviour of ants, and the solar system algorithm [12], which mimics the orbiting behaviour of celestial objects found in the solar system. These MAs have demonstrated considerable success across various applications due to their adaptability and robustness in exploring complex and large search spaces.

One of the key factors influencing the performance of population-based MAs is the initialization of the population. The way the initial population is generated can significantly impact the algorithm's convergence speed, accuracy, and ability to explore the solution space effectively. Traditionally, random number generators, which use uniform probability distributions, have been the most commonly employed method for population initialization. However, the limitations of random initialization, such

TACC'2024, The 4th Tunisian-Algerian Conference on applied Computing, December 17-18, 2024, Constantine, Algeria

*Corresponding author.

†These authors contributed equally.

ferhat.aridj@univ-ouargla.dz (F. Aridj); zitouni.farouq@univ-ouargla.dz (Z. Farouq); limane.abdelhadi@univ-ouargla.dz (L. Abdelhadi); lakbichi.rihab@univ-ouargla.dz (L. Rihab); harous@sharjah.ac.ae (S. Harous)

as poor diversity and uneven coverage of the search space, have driven researchers to investigate alternative initialization strategies [13]. Recent studies have explored various approaches to improve population initialization, such as using different probability distributions (e.g., Beta, Rayleigh, Exponential) [14], quasi-random sequences (e.g., Halton, Sobol, Faure) [15], and chaos-based methods [16]. Additionally, hybrid techniques combining MAs with other optimization methods have been proposed to enhance the initialization process and improve the overall performance [17]. These methods aim to ensure better diversity and coverage of the search space, which are critical for avoiding premature convergence and ensuring that MAs can explore a wide range of possible solutions.

In this paper, we examine the impact of nine distinct initialization methods on the performance of AHO [18]. These methods include six probability distributions: Beta, Normal, Logarithmic Normal, Exponential, Rayleigh, and Weibull distributions [14], as well as three low-discrepancy sequences: Halton, Faure, and Sobol [15]. In addition, we analyze how varying population sizes and iteration counts affect the optimizer's efficiency. The findings reveal that AHO's performance is significantly influenced by the choice of initialization scheme. Based on these findings, we propose optimal population sizes, iteration counts, and initialization methods for enhancing AHO's effectiveness across different benchmark functions.

The paper is structured as follows: Section 2 provides a brief overview of related work in the literature. Section 3 discusses background information on pseudo-random number generators, probability distributions, and the Archerfish Hunting Optimizer used in our experiments, which are presented in Section 4. The paper concludes in Section 5 with suggestions for future work to expand on the findings of this study.

2. Related Work

Initialization is critical in implementing MAs, significantly influencing their efficiency and performance in finding optimal solutions to complex optimization problems. Traditional approaches often rely on random number generators to create initial populations. However, the limitations of random initialization, such as uneven coverage of the search space, leading to premature convergence, and poor performance in high-dimensional problems, are well-documented in the literature, prompting researchers to explore alternative probability distributions to enhance algorithm performance [19, 13].

To overcome these limitations, numerous studies have investigated the effects of different probabilistic distributions as alternative initialization methods. For instance, in a comprehensive study, conducted in [14], 22 different initialization methods based on distributions such as Beta, Rayleigh, and Exponential were compared across five MAs: differential evolution [20], particle swarm optimization [21], cuckoo search [22], artificial bee colony [23], and genetic algorithm [9]. The results indicated that some of these distributions could significantly improve the accuracy and convergence of these optimizers compared to traditional random initialization.

Another notable work explored the significance of initialization methods on the performance of population-based MAs [24]. The authors examined the impact of population size, the number of iterations, and eleven different initialization techniques on ten widely used optimizers, including the bat algorithm [25], grey wolf optimizer [26], and whale optimization algorithm [27]. The study revealed that specific algorithms were sensitive to the choice of initialization methods, which influenced their convergence and accuracy across various test functions. Furthermore, the research underscored that the optimal population size and the number of iterations varied depending on the specific algorithm and problem, highlighting the importance of tailored initialization strategies to boost algorithmic performance.

In another work [15], researchers delved into the application of randomized low-discrepancy sequences, specifically the Halton, Sobol, and Faure sequences, for initializing particle swarms in the particle swarm optimization algorithm [28]. The study compared its performance using these low-discrepancy sequences with the traditional uniform initialization method that employs a pseudo-random number generator. The findings suggested that the use of randomized low-discrepancy sequences could enhance the algorithm's performance on benchmark problems. Moreover, the study indicated that different low-discrepancy sequences might have varying effects on the algorithm's performance, suggesting that the choice of sequence could influence optimization outcomes.

Finally, an article provided a comprehensive review of various initialization methods used in MAs to enhance their performance [29]. It categorized initialization schemes into several groups, including random numbers, quasi-random sequences, chaos theory, and probability distributions. The authors conducted experiments comparing ten different initialization methods, evaluating their impact on the performance of three MAs: bat algorithm [25], grey wolf optimizer [26], and butterfly optimization algorithm [30]. The experiments emphasized the importance of balancing population size, the diversity of the initial population, and the number of iterations as crucial factors in achieving optimal solutions in complex optimization problems.

3. Background

This section provides an overview of AHO and the various initialization methods adopted in this study. These foundational concepts are critical to understanding how these initialization techniques influence AHO's performance.

3.1. Archerfish Hunting Optimizer

AHO [18] is an MA inspired by the hunting strategies of archerfish, which capture prey by either shooting water droplets or leaping out of water. AHO translates these behaviours into an optimization algorithm where solutions are refined through iterations, focusing on efficient search space exploration and the avoidance of local optima. The algorithm operates with two key parameters: the attractiveness rate (ω) and the swapping angle (θ).

The algorithm begins by initializing a population of candidate solutions $\mathbf{x}_i$ within the search space boundaries using Equation 1. The symbols $\mathbf{r}_i$ represent D-dimensional vectors drawn from the uniform distribution. The vectors $\mathbf{ub}$ and $\mathbf{lb}$ denote the upper and lower bounds of the search space, respectively.

$$\mathbf{x}_i = \mathbf{r}_i \circ (\mathbf{ub} - \mathbf{lb}) + \mathbf{lb} \ , \ i \in \{1, \dots, N\} \tag{1}$$

The generation of new solutions during the search process is governed by three key phases: exploration, exploitation, and local optima avoidance. The exploration phase uses Equation 2; the exploitation phase refines solutions by employing Equation 3; and to avoid getting trapped in local optima, Equation 4 is utilized. The symbol α is a random number uniformly distributed between 0 and 1. The symbol $\mathbf{L}_D$ is a vector derived from the Lévy flight distribution. The symbol ϵ is a small number. The symbol b is a random number drawn from $\{0, 1\}$.

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{best}} + \exp\left(-\|\mathbf{x}_1 - \mathbf{x}_i\|_2^2\right) \times (\mathbf{x}_1 - \mathbf{x}_i) \tag{2}$$

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{best}} + \exp\left(-\|\mathbf{x}_2 - \mathbf{x}_i\|_2^2\right) \times (\mathbf{x}_2 - \mathbf{x}_i) \tag{3}$$

$$\mathbf{x}_{\text{new}} = \mathbf{x}_i + \alpha \times \mathbf{L}_D \tag{4}$$

$$\mathbf{x}_1 = \mathbf{x}_i + \epsilon \times \mathbf{1}_D + \mathbf{v}_1 \ , \ \mathbf{x}_2 = \mathbf{x}_i + \epsilon \times \mathbf{1}_D + \mathbf{v}_2$$

$$\mathbf{v}_1^{(j)} = \begin{cases} 0 & , \quad j \in \{1, \dots, D\} \setminus \{\sigma(1)\} \\ \omega \times \sin(2 \times \theta_0) & , \quad j = \sigma(1) \end{cases}$$

$$\mathbf{v}_2^{(j)} = \begin{cases} 0 & , \quad j \in \{1, \dots, D\} \setminus \{\sigma(1), \sigma(2)\} \\ \omega \times \sin(2 \times \theta_0) & , \quad j = \sigma(1) \\ \omega \times \sin^2(\theta_0) & , \quad j = \sigma(2) \end{cases}$$

$$\theta_0 = (-1)^b \times \alpha \times \pi$$

3.2. Probability Distributions

Usually, distributions can be used as initial population schemes, where each affects MAs' search behaviour differently. In the following sections, some common distributions are described.

3.2.1. Beta Distribution

It is defined over $(0, 1)$ with shape parameters a and b. It can be written as $X \sim \mathrm{Be}(a, b)$. Its probability density function (PDF) is given by Equation 5, where the symbol Γ represents the Gamma function. Also, its expected value and variance are provided in Equation 6.

$$f(x; a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1 - x)^{b-1} \tag{5}$$

$$\begin{cases} E[X] = \frac{a}{a+b} \\ \mathrm{Var}(X) = \frac{ab}{(a+b)(a+b+1)} \end{cases} \tag{6}$$

3.2.2. Uniform Distribution

It is defined over $[a, b]$. It can be written as $X \sim U(a, b)$. Its PDF is described by Equation 7, and its mean and variance are calculated using Equation 8.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } a \le x \le b \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$\begin{cases} E[X] = \frac{a+b}{2} \\ \mathrm{Var}(X) = \frac{(b-a)^2}{12} \end{cases} \tag{8}$$

3.2.3. Normal Distribution

It is characterized by mean μ and variance σ^2. It can be written as $X \sim N(\mu, \sigma^2)$. Its PDF is given by Equation 9, and its mean and variance are described in Equation 10.

$$f(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) \tag{9}$$

$$\begin{cases} E[X] = \mu \\ \mathrm{Var}(X) = \sigma^2 \end{cases} \tag{10}$$

3.2.4. Logarithmic Normal Distribution

It is defined as the logarithm of the variable following a normal distribution. It can be written as $\ln X \sim N(\mu, \sigma^2)$. Its PDF is given by Equation 11, and its mean and variance are specified in Equation 12.

$$f(x; \mu, \sigma^2) = \frac{1}{x\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\ln x - \mu)^2}{2\sigma^2}\right) \tag{11}$$

$$\begin{cases} E[X] = \exp\left(\mu + \frac{\sigma^2}{2}\right) \\ \mathrm{Var}(X) = \left(\exp(\sigma^2) - 1\right) \exp\left(2\mu + \sigma^2\right) \end{cases} \tag{12}$$

3.2.5. Exponential Distribution

It can be written as $X \sim \exp(\lambda)$. Its PDF is given by Equation 13, and its mean and variance are provided in Equation 14 ($\lambda > 0$).

$$f(x;\lambda) = \begin{cases} \lambda \exp(-\lambda x) & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases} \tag{13}$$

$$\begin{cases} E[X] = \frac{1}{\lambda} \\ \operatorname{Var}(X) = \frac{1}{\lambda^2} \end{cases} \tag{14}$$

3.2.6. Rayleigh Distribution

It can be written as $X \sim \text{Rayleigh}(\sigma)$. Its PDF is given by Equation 15, and its mean and variance are calculated using Equation 16.

$$f(x;\sigma) = \frac{x}{\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{15}$$

$$\begin{cases} E[X] = \sigma\sqrt{\frac{\pi}{2}} \\ \operatorname{Var}(X) = \frac{4-\pi}{2}\sigma^2 \end{cases} \tag{16}$$

3.2.7. Weibull Distribution

It generalizes several distributions with scale parameter λ and shape parameter k. It can be written as $X \sim Weibull(\lambda, k)$. Its PDF is given by Equation 17, and its mean and variance are given in Equation 18.

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} \exp\left(-\left(\frac{x}{\lambda}\right)^k\right) & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases} \tag{17}$$

$$\begin{cases} E[X] = \lambda\Gamma\left(1+\frac{1}{k}\right) \\ \operatorname{Var}(X) = \lambda^2\left[\Gamma\left(1+\frac{2}{k}\right) - \Gamma\left(1+\frac{1}{k}\right)^2\right] \end{cases} \tag{18}$$

3.3. Low-Discrepancy Sequences

Quasi-random number generators are known to generate sequences that have low discrepancy, which makes them powerful tools for improving optimization algorithms. Low-discrepancy sequences, such as Van der Corput [31], Sobol [31], Faure [32], and Halton [33], are particularly useful in uniformly covering the search space, which enhances the initialization process in MAs. This section will focus on three common low-discrepancy sequences – Halton, Faure, and Sobol – which have shown significant promise in initialization schemes for MAs.

3.3.1. Randomized Halton Sequence

The Halton Sequence [34, 31] extends the Van der Corput sequence to multiple dimensions. The Van der Corput sequence in a given base b is defined by representing a non-negative integer n in base-b form, as shown in Equation 19.

$$n = a_0 + a_1 b + a_2 b^2 + \cdots + a_m b^m \tag{19}$$

, where a_i are the digits in base b, and m is the largest integer such that $b^m \leq n$. The Van der Corput sequence maps n to a fraction by reversing the digits of the base-b expansion, according to Equation 20.

$$\varphi_b(n) = \frac{a_0}{b} + \frac{a_1}{b^2} + \cdots + \frac{a_m}{b^{m+1}} \tag{20}$$

For higher dimensions, the Halton sequence uses different prime numbers $p_1, p_2, \ldots, p_d$ as bases, as described in Equation 21. To address the deterministic nature of the Halton sequence, small perturbations such as Gaussian noise are introduced, which ensures variability between multiple algorithm runs.

$$\mathbf{x}_n = (\varphi_{p_1}(n), \varphi_{p_2}(n), \ldots, \varphi_{p_d}(n)) \tag{21}$$

, where each $\varphi_{p_i}(n)$ is the radical inverse function in base p_i.

3.3.2. Randomized Faure Sequence

The Faure sequence [34, 32] is a permutation of the Halton sequence with a single base m, chosen as the smallest prime number greater than or equal to the number of dimensions. The representation of a point C_n is given by Equation 22. The Faure sequence improves stratification by applying a specific permutation to the coefficients. The recursive permutation for the Faure sequence is outlined in Equation 23.

$$C_n = b_0^{m-1} + b_1^{m-2} + \cdots + b_r^{m-(r+1)} \tag{22}$$

$$b_j \equiv \sum_{i \geq j}^{r} \binom{i}{j} a_i \mod m \tag{23}$$

We also apply randomization to the Faure sequence by introducing small Gaussian noise to the coordinates. This randomization ensures that the Faure sequence can be used in randomized MAs, providing better performance compared to pseudo-random numbers.

3.3.3. Randomized Sobol Sequence

The Sobol sequence [34, 33] is another popular low-discrepancy sequence based on linear recurrence relations over a finite field. The n-th element of the Sobol sequence in dimension j, denoted by $x_n^{(j)}$, is computed using the recurrence relation defined in Equation 24.

$$x_n^{(j)} = v_1^{(j)} n_1 \oplus v_2^{(j)} n_2 \oplus \ldots \oplus v_w^{(j)} n_w \tag{24}$$

, where $v_i^{(j)}$ are known as direction numbers, and the symbol $\oplus$ denotes the bit-wise XOR operation.

We also introduce randomization by adding Gaussian noise to the generated points to enhance exploratory capabilities while maintaining the low discrepancy properties of the Sobol sequence.

4. Experiments, and Discussion

To examine how different initialization methods influence the performance of AHO, we conducted a series of experiments. The key objective is to assess how varying the population size (NP) and the maximum number of iterations (T) affects the algorithm's efficiency, with the ultimate goal of determining the optimal values for NP and T that produce the best outcomes. The total number of function evaluations (NFEs) is consistently set at 600,000, while the number of iterations is adjusted based on the chosen population size. Consequently, larger NP values corresponded to fewer iterations, while smaller NP values allowed for more iterations. The NP and T combinations tested include $C_1 = (10, 60000)$, $C_2 = (30, 20000)$, $C_3 = (60, 10000)$, and $C_4 = (100, 6000)$, with the goal of identifying the combination that delivers the best performance for AHO.

The experiments were conducted on a machine equipped with an Intel® Core™ i7-8750H CPU, operating at a base frequency of 2.20 GHz with a boost frequency of 2.21 GHz. The system is equipped with 32 GB of RAM and runs Windows 11. All computational analyses were performed using Matlab R2023a. Each test function was evaluated with 30 independent runs to reduce variance.

We evaluated the performance of AHO across functions F_1, F_2, F_6, and F_9 in dimensions 10 and 20 (Not all twelve benchmark functions were included due to space constraints). These functions were specifically chosen to represent different categories of benchmark functions – unimodal, multimodal, hybrid, and composition. Table 1 provides a detailed summary of the Best, Worst, Mean, Standard Deviation (STD), and mean execution time (ET) values obtained from 30 independent runs. The STD was computed based on the optimal solutions of the test functions, as defined in the CEC 2022 benchmark[1]. Finally, the table presents the statistical analysis performed using Friedman's test, showing that the best performance for both the 10-dimensional and 20-dimensional cases was achieved when NP = 30 and T = 20000.

Table 1
Impact of varying the population size and the number of iterations on the obtained performance.

		$D = 10$				$D = 20$			
		C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_4
F_1	**Best**	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02
	Worst	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.003e+02
	Mean	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.001e+02
	STD	4.792e-11	2.462e-12	5.452e-12	7.772e-11	8.542e-03	8.357e-03	2.056e-02	8.288e-02
	ET	1.877e+01	1.739e+01	1.662e+01	2.082e+01	2.431e+01	1.857e+01	1.747e+01	2.107e+01
F_2	**Best**	4.000e+02	4.000e+02	4.000e+02	4.000e+02	4.000e+02	4.000e+02	4.000e+02	4.001e+02
	Worst	4.708e+02	4.089e+02	4.708e+02	4.708e+02	4.491e+02	4.491e+02	4.491e+02	4.709e+02
	Mean	4.051e+02	4.047e+02	4.100e+02	4.110e+02	4.316e+02	4.289e+02	4.234e+02	4.407e+02
	STD	1.395e+01	6.242e+00	1.975e+01	2.341e+01	3.942e+01	3.779e+01	3.354e+01	4.637e+01
	ET	1.860e+01	1.767e+01	1.667e+01	2.099e+01	2.565e+01	1.898e+01	1.801e+01	2.049e+01
F_6	**Best**	1.831e+03	1.838e+03	1.835e+03	1.826e+03	1.902e+03	1.942e+03	1.968e+03	1.887e+03
	Worst	8.060e+03	8.013e+03	8.068e+03	8.013e+03	2.042e+04	2.034e+04	1.999e+04	1.710e+04
	Mean	4.280e+03	3.729e+03	3.858e+03	4.107e+03	7.066e+03	6.448e+03	8.222e+03	6.727e+03
	STD	3.467e+03	2.747e+03	2.872e+03	3.295e+03	8.414e+03	7.000e+03	8.333e+03	7.314e+03
	ET	1.907e+01	1.741e+01	1.693e+01	2.244e+01	2.790e+01	1.947e+01	1.807e+01	2.078e+01
F_9	**Best**	2.529e+03	2.529e+03	2.529e+03	2.529e+03	2.481e+03	2.481e+03	2.481e+03	2.481e+03
	Worst	2.676e+03	2.676e+03	2.676e+03	2.529e+03	2.481e+03	2.481e+03	2.481e+03	2.481e+03
	Mean	2.544e+03	2.534e+03	2.534e+03	2.529e+03	2.481e+03	2.481e+03	2.481e+03	2.481e+03
	STD	2.522e+02	2.397e+02	2.397e+02	2.332e+02	1.839e+02	1.839e+02	1.839e+02	1.839e+02
	ET	2.116e+01	1.983e+01	1.942e+01	2.441e+01	3.160e+01	2.607e+01	2.195e+01	2.642e+01
Mean Rank		3.2500	**1.3750**	2.3750	3.0000	2.0000	**1.7500**	2.7500	3.5000

After determining the optimal values for NP and T, we assess the effectiveness of various probability distributions and low-discrepancy sequences for initializing the population in AHO. Specifically, we tested six probability distributions – Beta, Normal, Logarithmic Normal, Exponential, Rayleigh, and Weibull – as well as three low-discrepancy sequences: Halton, Faure, and Sobol. Each distribution and sequence is used to generate the initial populations, and the performance is compared to the original algorithm, which utilizes a uniform distribution for initialization. We provide an overview of the initialization methods and their parameter settings in Table 2. Additionally, the results derived from the comparison of these methods are systematically presented in Tables 3 and 4.

We give an in-depth analysis of the performance of AHO on the CEC 2022 benchmark functions, using various initialization schemes, across two dimensions: $D = 10$ and $D = 20$. First, the discussion will be focused on the type of test functions (unimodal, basic, hybrid, and composition). Then, an analysis of the Friedman statistical test will be provided. Finally, a general overview of some key observations is given.

For the unimodal function F_1 at dimension $D = 10$, initialization schemes such as D_2, D_3, D_4, D_5, and D_6 exhibited very poor performance, with significantly high mean values and large standard deviations, reflecting instability in the performance. This contrasts with their performance at $D = 20$, where the results were more consistent across all initialization methods. At $D = 20$, the best, worst, and mean values were identical at 300 for all methods, indicating that this unimodal function does not effectively distinguish between different initialization techniques when applied in the context of AHO. This suggests that for unimodal functions, the choice of initialization method becomes more

[1]https://github.com/P-N-Suganthan/2022-SO-BO

Table 2
Overview of the used probability distributions and low-discrepancy sequences

Initialization Method	Parameter	Value
Random (D_0)	–	–
Beta (D_1)	(a, b)	$(3, 2)$
Normal Distribution (D_2)	(μ, σ)	$(0, 1)$
Logarithmic Normal (D_3)	(μ, σ)	$(0, 0.5)$
Exponential (D_4)	λ	0.5
Rayleigh (D_5)	σ	0.4
Weibull (D_6)	(λ, k)	$(1, 1)$
Sobol (S_1)	–	–
Halton (S_2)	–	–
Faure (S_3)	–	–

critical at lower dimensions, where instability in certain methods can negatively impact performance. However, at higher dimensions, this sensitivity diminishes, leading to more uniform outcomes across various methods. When analyzing the basic functions (F_2 to F_5) at $D = 10$, the results showed more variability. Initialization methods like D_2, D_3, D_4, and D_5 produced unstable results, characterized by high standard deviation. However, the S_2 and S_3 were the most stable, showing the lowest standard deviations across these functions. The D_1 also demonstrated competitive performance, closely matching the results of D_0. At $D = 20$, the results followed the same general pattern, though the average values and standard deviations increased slightly, with S_3 showing the best performance across three of the four functions, while D_5 remained inconsistent. For the hybrid functions (F_6 to F_8), the majority of methods displayed similar performance, but D_2 was particularly unstable and performed poorly across both dimensions. At $D = 20$, methods like D_2, D_3, D_4, D_5, and D_6 delivered the worst results, especially on function F6. Meanwhile, S_2, S_3, and D_1 were more stable and provided better results. S_1 and S_3 remained the top-performing methods, confirming their effectiveness in hybrid functions. These findings indicate that for hybrid functions, the careful selection of an initialization method is crucial. The probability distributions, with the exception of D_1, were generally ineffective for this category of functions. In contrast, low-discrepancy sequences, consistently performed well, suggesting that they are better suited for hybrid optimization problems. In the composition functions (F_9 to F_{12}), S_2 consistently outperformed other methods at both $D = 10$ and $D = 20$. This performance highlights the effectiveness of low-discrepancy sequences in efficiently navigating complex search spaces, which is essential for composition functions characterized by their intricate landscapes and multiple optima.

The Friedman test (last columns of Tables 3 and 4), used to rank the initialization methods across all functions, further highlighted the statistical superiority of S_2 and D_1 in both dimensions. These methods consistently demonstrated stability and efficiency, yielding optimal results with low standard deviations. S_1 also outperformed D_0 at $D = 10$, while S_3 showed better performance than D_0 at $D = 20$, though it did not perform as well at $D = 10$. Most initialization methods outperformed D_0 at $D = 20$, with S_2, S_3, and D_1 being the most effective. At $D = 10$, only three methods outperformed D_0, with S_2, S_1, and D_1 leading the way.

Finally, the results from the Friedman test rankings of execution times for different initialization methods across the dimensions $D = 10$ and $D = 20$, as presented in Table 5, highlight significant differences in computational efficiency. At $D = 10$, methods D_0 and D_1 ranked among the slowest with scores of 6.5000, indicating high computational overhead, while methods D_3 and D_4 exhibited relatively better execution times, although they still performed poorly overall. In contrast, low-discrepancy sequence methods such as S_1, S_2, and S_3 displayed more balanced execution times, with S_2 achieving a score of 5.0833. At $D = 20$, the efficiency landscape shifted, with S_1 ranking the lowest at 6.8333, while D_2 and D_0 also exhibited poor performance. Notably, methods D_4, D_5, and D_6 provided the shortest execution times but did not perform well on this dimension. Overall, low-discrepancy sequences consistently outperformed probabilistic methods in execution time, reinforcing their suitability for optimization tasks where both solution quality and computational efficiency are critical.

Table 3
AHO's performance comparison for CEC 2022 functions with different initialization schemes ($D = 10$).

		Initialization Methods									
		D_0	D_1	D_2	D_3	D_4	D_5	D_6	S_1	S_2	S_3
F_1	**Best**	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02
	Worst	3.000e+02	3.000e+02	5.893e+08	1.080e+10	1.302e+10	2.602e+06	8.363e+10	3.000e+02	3.000e+02	3.000e+02
	Mean	3.000e+02	3.000e+02	3.884e+07	3.599e+08	4.582e+08	1.104e+05	3.532e+09	3.000e+02	3.000e+02	3.000e+02
	STD	2.403e-12	**2.144e-12**	1.529e+08	2.005e+09	2.421e+09	4.970e+05	1.580e+10	2.744e-12	2.431e-12	2.580e-12
	ET	1.710e+01	1.681e+01	1.688e+01	1.556e+01	1.538e+01	1.611e+01	1.622e+01	1.547e+01	1.545e+01	1.551e+01
F_2	**Best**	4.000e+02	4.000e+02	4.000e+02	4.000e+02	4.000e+02	4.000e+02	4.000e+02	4.001e+02	4.000e+02	4.000e+02
	Worst	4.708e+02	4.708e+02	2.102e+04	4.455e+03	4.708e+02	7.669e+03	4.089e+02	4.041e+02	4.000e+02	4.089e+02
	Mean	4.105e+02	4.184e+02	1.874e+03	5.505e+02	4.313e+02	8.108e+02	4.029e+02	4.011e+02	4.000e+02	4.051e+02
	STD	2.334e+01	3.500e+01	4.659e+03	7.535e+02	4.578e+01	1.611e+03	4.913e+00	2.020e+00	**9.251e-03**	6.624e+00
	ET	1.582e+01	1.579e+01	1.561e+01	1.478e+01	1.499e+01	1.563e+01	1.572e+01	1.596e+01	1.561e+01	1.591e+01
F_3	**Best**	6.429e+02	6.337e+02	6.347e+02	6.331e+02	6.676e+02	6.283e+02	6.307e+02	6.599e+02	6.592e+02	6.328e+02
	Worst	6.794e+02	6.914e+02	6.868e+02	6.813e+02	7.184e+02	6.961e+02	6.959e+02	6.604e+02	6.602e+02	6.805e+02
	Mean	6.591e+02	6.601e+02	6.530e+02	6.589e+02	6.922e+02	6.532e+02	6.635e+02	6.599e+02	6.592e+02	6.562e+02
	STD	6.086e+01	6.263e+01	5.567e+01	6.162e+01	9.488e+01	**5.560e+01**	6.765e+01	6.097e+01	6.025e+01	5.806e+01
	ET	2.819e+01	2.844e+01	2.699e+01	2.680e+01	2.603e+01	2.731e+01	2.785e+01	2.759e+01	2.773e+01	2.918e+01
F_4	**Best**	8.209e+02	8.119e+02	8.209e+02	8.179e+02	8.229e+02	8.139e+02	8.279e+02	8.378e+02	8.517e+02	8.149e+02
	Worst	8.900e+02	8.776e+02	9.234e+02	9.363e+02	9.573e+02	8.935e+02	9.592e+02	8.378e+02	8.517e+02	8.925e+02
	Mean	8.552e+02	8.480e+02	8.767e+02	8.707e+02	8.984e+02	8.406e+02	8.766e+02	8.378e+02	8.517e+02	8.544e+02
	STD	5.819e+01	5.075e+01	8.336e+01	7.758e+01	1.037e+02	4.591e+01	8.464e+01	**3.845e+01**	5.262e+01	5.906e+01
	ET	2.795e+01	2.942e+01	2.862e+01	2.904e+01	2.913e+01	2.931e+01	2.825e+01	2.860e+01	2.859e+01	2.777e+01
F_5	**Best**	1.674e+03	1.222e+03	1.323e+03	1.606e+03	2.511e+03	1.326e+03	1.667e+03	1.594e+03	1.538e+03	1.321e+03
	Worst	4.555e+03	2.925e+03	4.514e+03	4.394e+03	5.201e+03	3.187e+03	3.965e+03	1.611e+03	1.538e+03	3.432e+03
	Mean	2.338e+03	1.795e+03	2.526e+03	2.878e+03	4.011e+03	1.881e+03	2.714e+03	1.601e+03	1.538e+03	2.197e+03
	STD	1.567e+03	9.791e+02	1.793e+03	2.118e+03	3.246e+03	1.069e+03	1.991e+03	7.127e+02	**6.489e+02**	1.425e+03
	ET	2.964e+01	2.949e+01	2.912e+01	2.822e+01	2.853e+01	2.915e+01	3.005e+01	2.903e+01	2.943e+01	2.909e+01
F_6	**Best**	1.831e+03	1.815e+03	1.823e+03	1.810e+03	1.889e+03	1.857e+03	1.819e+03	2.355e+03	1.973e+03	1.825e+03
	Worst	8.059e+03	7.673e+03	7.999e+08	7.995e+03	5.874e+03	7.010e+03	8.061e+03	2.553e+03	2.033e+03	8.002e+03
	Mean	4.098e+03	2.920e+03	2.667e+07	3.407e+03	3.226e+03	3.141e+03	4.046e+03	2.421e+03	2.005e+03	3.627e+03
	STD	3.259e+03	1.965e+03	1.485e+08	2.150e+03	1.922e+03	1.831e+03	3.246e+03	6.327e+02	**2.086e+02**	2.666e+03
	ET	2.899e+01	2.938e+01	2.942e+01	2.887e+01	2.857e+01	2.916e+01	2.915e+01	2.918e+01	2.824e+01	2.894e+01
F_7	**Best**	2.048e+03	2.049e+03	2.038e+03	2.023e+03	2.076e+03	2.045e+03	2.050e+03	2.158e+03	2.132e+03	2.045e+03
	Worst	2.250e+03	2.289e+03	2.266e+03	2.298e+03	2.265e+03	2.273e+03	2.250e+03	2.176e+03	2.231e+03	2.275e+03
	Mean	2.126e+03	2.128e+03	2.173e+03	2.155e+03	2.201e+03	2.150e+03	2.160e+03	2.164e+03	2.227e+03	2.141e+03
	STD	**1.397e+02**	**1.397e+02**	1.853e+02	1.766e+02	2.111e+02	1.654e+02	1.714e+02	1.669e+02	2.319e+02	1.537e+02
	ET	3.365e+01	3.283e+01	3.311e+01	3.277e+01	3.374e+01	3.414e+01	3.280e+01	3.291e+01	3.343e+01	3.393e+01
F_8	**Best**	2.227e+03	2.228e+03	2.221e+03	2.221e+03	2.346e+03	2.221e+03	2.222e+03	2.351e+03	2.602e+03	2.223e+03
	Worst	2.685e+03	2.534e+03	2.621e+03	2.641e+03	2.646e+03	2.647e+03	2.585e+03	2.358e+03	2.604e+03	2.621e+03
	Mean	2.393e+03	2.431e+03	2.308e+03	2.290e+03	2.470e+03	2.437e+03	2.330e+03	2.355e+03	2.603e+03	2.380e+03
	STD	2.423e+02	2.475e+02	1.578e+02	**1.571e+02**	2.869e+02	2.807e+02	1.768e+02	1.579e+02	4.096e+02	2.234e+02
	ET	3.487e+01	3.388e+01	3.451e+01	3.485e+01	3.579e+01	3.564e+01	3.420e+01	3.490e+01	3.545e+01	3.417e+01
F_9	**Best**	2.529e+03	2.529e+03	2.529e+03	2.529e+03	2.400e+03	2.300e+03	2.529e+03	2.529e+03	2.529e+03	2.529e+03
	Worst	2.529e+03	2.529e+03	3.638e+03	2.676e+03	2.529e+03	3.261e+03	4.046e+03	2.529e+03	2.529e+03	2.676e+03
	Mean	2.529e+03	2.529e+03	2.617e+03	2.539e+03	2.521e+03	2.546e+03	2.611e+03	2.529e+03	2.529e+03	2.534e+03
	STD	2.332e+02	2.332e+02	4.151e+02	2.460e+02	**2.268e+02**	2.874e+02	4.343e+02	2.332e+02	2.332e+02	2.397e+02
	ET	3.193e+01	3.059e+01	3.137e+01	2.963e+01	2.948e+01	3.095e+01	3.136e+01	2.965e+01	2.962e+01	3.100e+01
F_{10}	**Best**	2.501e+03	2.501e+03	2.501e+03	2.501e+03	2.501e+03	2.501e+03	2.501e+03	2.501e+03	2.631e+03	2.501e+03
	Worst	3.904e+03	4.151e+03	4.371e+03	4.361e+03	4.325e+03	3.825e+03	4.545e+03	2.502e+03	4.553e+03	4.234e+03
	Mean	2.946e+03	2.764e+03	3.516e+03	3.361e+03	3.584e+03	2.816e+03	3.404e+03	2.502e+03	4.483e+03	3.193e+03
	STD	7.682e+02	6.370e+02	1.220e+03	1.128e+03	1.333e+03	5.984e+02	1.188e+03	1.034e+02	2.147e+03	1.018e+03
	ET	2.949e+01	3.088e+01	2.956e+01	3.022e+01	3.006e+01	2.959e+01	3.230e+01	**2.956e+01**	3.085e+01	3.069e+01
F_{11}	**Best**	2.600e+03	2.600e+03	2.600e+03	2.600e+03	2.985e+03	2.600e+03	2.600e+03	2.600e+03	2.900e+03	2.600e+03
	Worst	4.857e+03	4.636e+03	5.465e+03	6.043e+03	6.527e+03	4.580e+03	6.697e+03	2.600e+03	4.471e+03	3.916e+03
	Mean	3.229e+03	3.242e+03	3.456e+03	3.313e+03	3.676e+03	2.962e+03	3.278e+03	2.600e+03	3.709e+03	2.851e+03
	STD	8.955e+02	8.097e+02	1.233e+03	1.188e+03	1.242e+03	6.207e+02	1.212e+03	**2.506e-12**	1.318e+03	4.823e+02
	ET	3.134e+01	3.272e+01	3.125e+01	3.336e+01	3.391e+01	3.170e+01	3.215e+01	3.277e+01	3.321e+01	3.392e+01
F_{12}	**Best**	2.916e+03	2.903e+03	2.915e+03	2.900e+03	2.869e+03	2.898e+03	2.882e+03	2.990e+03	3.113e+03	2.880e+03
	Worst	3.351e+03	3.265e+03	3.290e+03	2.900e+03	3.333e+03	3.299e+03	3.317e+03	3.185e+03	3.113e+03	3.175e+03
	Mean	3.036e+03	3.034e+03	3.019e+03	2.900e+03	3.118e+03	3.051e+03	3.052e+03	3.089e+03	3.113e+03	2.994e+03
	STD	3.544e+02	3.516e+02	3.363e+02	**2.034e+02**	4.462e+02	3.758e+02	3.756e+02	4.026e+02	4.204e+02	3.099e+02
	ET	3.447e+01	3.290e+01	3.395e+01	3.393e+01	3.262e+01	3.265e+01	3.276e+01	3.321e+01	3.370e+01	3.397e+01
Mean Rank		4.5833	4.2500	6.8333	6.0833	8.0833	4.7500	7.2500	**3.2500**	5.5833	4.3333

5. Conclusion and Perspectives

The research presented in this paper contributes to the growing body of knowledge on initialization schemes for MAs, an area that has yet to be fully explored in the optimization literature. While numerous MAs have been developed and refined over the years, much less attention has been given to the initialization phase, which significantly impacts the overall performance of these algorithms. Most researchers continue to rely on traditional random number generators for initialization, despite

Table 4
AHO's performance comparison for CEC 2022 functions with different initialization schemes ($D = 20$).

		Initialization Methods									
		D_0	D_1	D_2	D_3	D_4	D_5	D_6	S_1	S_2	S_3
F_1	**Best**	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02
	Worst	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02
	Mean	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02	3.000e+02
	STD	1.025e-02	6.507e-03	9.284e-03	9.673e-03	**3.336e-03**	7.448e-03	8.858e-03	3.965e-03	2.557e-03	1.041e-02
	ET	3.030e+01	3.069e+01	3.086e+01	3.019e+01	3.070e+01	3.155e+01	3.136e+01	3.096e+01	3.098e+01	3.018e+01
F_2	**Best**	4.000e+02	4.000e+02	4.000e+02	4.000e+02	4.000e+02	4.000e+02	4.000e+02	4.491e+02	4.491e+02	4.000e+02
	Worst	4.491e+02	4.491e+02	4.491e+02	4.491e+02	4.718e+02	7.436e+03	4.491e+02	4.491e+02	4.491e+02	4.727e+02
	Mean	4.320e+02	4.364e+02	4.357e+02	4.231e+02	4.449e+02	6.637e+02	4.279e+02	4.491e+02	4.491e+02	4.263e+02
	STD	3.960e+01	4.184e+01	4.228e+01	**3.354e+01**	4.835e+01	1.307e+03	3.670e+01	4.992e+01	4.992e+01	3.671e+01
	ET	3.056e+01	3.058e+01	3.041e+01	3.096e+01	2.963e+01	3.006e+01	3.045e+01	2.999e+01	3.036e+01	3.132e+01
F_3	**Best**	6.468e+02	6.602e+02	6.484e+02	6.471e+02	6.838e+02	6.399e+02	6.509e+02	6.716e+02	6.658e+02	6.560e+02
	Worst	6.956e+02	6.993e+02	6.940e+02	7.069e+02	7.291e+02	6.848e+02	6.946e+02	6.722e+02	6.666e+02	6.984e+02
	Mean	6.701e+02	6.756e+02	6.684e+02	6.755e+02	7.090e+02	6.695e+02	6.695e+02	6.718e+02	6.665e+02	6.704e+02
	STD	7.222e+01	7.745e+01	7.078e+01	7.801e+01	1.114e+02	7.149e+01	7.135e+01	7.304e+01	**6.759e+01**	7.257e+01
	ET	3.492e+01	3.599e+01	3.579e+01	3.444e+01	3.439e+01	3.486e+01	3.436e+01	3.462e+01	3.457e+01	3.442e+01
F_4	**Best**	9.025e+02	8.826e+02	8.998e+02	9.055e+02	9.900e+02	8.826e+02	8.776e+02	9.383e+02	8.876e+02	8.836e+02
	Worst	1.044e+03	9.691e+02	1.093e+03	1.158e+03	1.133e+03	9.980e+02	1.106e+03	9.393e+02	8.945e+02	1.003e+03
	Mean	9.481e+02	9.278e+02	9.859e+02	1.006e+03	1.073e+03	9.241e+02	9.695e+02	9.383e+02	8.941e+02	9.491e+02
	STD	1.563e+02	1.319e+02	1.946e+02	2.226e+02	2.803e+02	1.288e+02	1.810e+02	1.407e+02	**9.568e+01**	1.550e+02
	ET	3.228e+01	3.028e+01	3.137e+01	3.118e+01	3.063e+01	2.985e+01	3.106e+01	3.308e+01	3.173e+01	3.089e+01
F_5	**Best**	2.867e+03	2.416e+03	3.119e+03	2.762e+03	5.240e+03	2.117e+03	3.150e+03	3.032e+03	2.390e+03	2.981e+03
	Worst	5.959e+03	4.425e+03	7.872e+03	8.900e+03	1.916e+04	5.384e+03	7.895e+03	3.099e+03	2.430e+03	7.221e+03
	Mean	4.733e+03	3.294e+03	4.777e+03	5.268e+03	7.352e+03	3.616e+03	5.587e+03	3.064e+03	2.413e+03	4.382e+03
	STD	3.976e+03	2.506e+03	4.100e+03	4.657e+03	6.977e+03	2.850e+03	4.949e+03	2.201e+03	**1.539e+03**	3.689e+03
	ET	3.100e+01	3.181e+01	3.233e+01	3.056e+01	3.129e+01	3.191e+01	3.109e+01	3.137e+01	3.178e+01	3.291e+01
F_6	**Best**	1.878e+03	1.932e+03	2.040e+03	2.110e+03	1.853e+03	1.863e+03	2.006e+03	4.681e+03	3.471e+03	1.909e+03
	Worst	2.044e+04	1.477e+04	1.882e+10	1.042e+10	3.074e+10	8.670e+09	2.695e+10	6.138e+03	4.936e+03	2.043e+04
	Mean	1.124e+04	4.355e+03	6.274e+08	3.472e+08	5.613e+09	7.325e+08	2.977e+09	5.389e+03	3.952e+03	8.193e+03
	STD	1.189e+04	4.070e+03	3.495e+09	1.934e+09	1.234e+10	2.210e+09	8.484e+09	3.676e+03	**2.212e+03**	8.828e+03
	ET	3.216e+01	2.958e+01	3.023e+01	3.184e+01	2.998e+01	3.006e+01	3.151e+01	3.063e+01	3.196e+01	3.014e+01
F_7	**Best**	2.093e+03	2.122e+03	2.086e+03	2.139e+03	2.219e+03	2.136e+03	2.087e+03	2.362e+03	2.438e+03	2.098e+03
	Worst	2.561e+03	2.747e+03	2.720e+03	2.779e+03	2.722e+03	2.576e+03	2.507e+03	2.373e+03	2.449e+03	2.627e+03
	Mean	2.330e+03	2.289e+03	2.401e+03	2.404e+03	2.451e+03	2.286e+03	2.323e+03	2.371e+03	2.446e+03	2.337e+03
	STD	3.562e+02	**3.229e+02**	4.497e+02	4.356e+02	4.753e+02	3.235e+02	3.498e+02	3.774e+02	4.535e+02	3.742e+02
	ET	3.795e+01	3.910e+01	3.746e+01	3.845e+01	3.922e+01	3.842e+01	3.790e+01	3.906e+01	3.790e+01	3.910e+01
F_8	**Best**	2.293e+03	2.253e+03	2.225e+03	2.236e+03	2.248e+03	2.234e+03	2.235e+03	2.354e+03	2.238e+03	2.253e+03
	Worst	3.011e+03	3.018e+03	2.878e+03	2.894e+03	2.892e+03	3.024e+03	1.963e+05	2.370e+03	2.251e+03	3.014e+03
	Mean	2.620e+03	2.665e+03	2.460e+03	2.543e+03	2.620e+03	2.593e+03	8.979e+03	2.360e+03	2.243e+03	2.554e+03
	STD	4.627e+02	5.031e+02	3.271e+02	4.073e+02	4.558e+02	4.323e+02	3.604e+04	1.627e+02	**4.372e+01**	3.996e+02
	ET	3.993e+01	4.124e+01	4.122e+01	4.065e+01	4.229e+01	4.013e+01	4.103e+01	4.251e+01	4.154e+01	4.132e+01
F_9	**Best**	2.481e+03	2.481e+03	2.481e+03	2.481e+03	2.481e+03	2.400e+03	2.481e+03	2.481e+03	2.481e+03	2.481e+03
	Worst	2.481e+03	2.481e+03	5.236e+03	2.481e+03	5.308e+03	3.716e+03	5.634e+03	2.481e+03	2.481e+03	2.481e+03
	Mean	2.481e+03	2.481e+03	2.617e+03	2.481e+03	2.669e+03	2.520e+03	2.586e+03	2.481e+03	2.481e+03	2.481e+03
	STD	1.839e+02	1.839e+02	6.386e+02	1.839e+02	8.098e+02	3.182e+02	6.450e+02	1.839e+02	1.839e+02	1.839e+02
	ET	3.741e+01	3.660e+01	3.719e+01	3.735e+01	3.649e+01	3.573e+01	3.672e+01	3.692e+01	3.580e+01	3.547e+01
F_{10}	**Best**	2.501e+03	3.987e+03	2.849e+03	3.894e+03	4.223e+03	2.501e+03	2.501e+03	4.580e+03	5.682e+03	2.501e+03
	Worst	6.271e+03	6.422e+03	5.851e+03	6.539e+03	6.367e+03	6.285e+03	6.093e+03	5.078e+03	5.682e+03	6.030e+03
	Mean	4.869e+03	5.268e+03	5.169e+03	5.152e+03	5.422e+03	4.961e+03	5.141e+03	4.798e+03	5.682e+03	5.210e+03
	STD	2.688e+03	2.970e+03	2.877e+03	2.861e+03	3.140e+03	2.731e+03	2.884e+03	**2.445e+03**	3.338e+03	2.944e+03
	ET	3.471e+01	3.462e+01	3.521e+01	3.501e+01	3.368e+01	3.477e+01	3.511e+01	3.489e+01	3.418e+01	3.434e+01
F_{11}	**Best**	2.600e+03	2.900e+03	2.600e+03	2.600e+03	2.900e+03	2.600e+03	2.600e+03	2.900e+03	2.900e+03	2.600e+03
	Worst	9.703e+03	8.531e+03	3.000e+03	3.000e+03	3.000e+03	3.000e+03	1.153e+04	2.900e+03	6.712e+03	8.582e+03
	Mean	4.000e+03	3.215e+03	2.883e+03	2.901e+03	2.903e+03	2.883e+03	3.613e+03	2.900e+03	3.044e+03	3.273e+03
	STD	2.585e+03	1.266e+03	3.057e+02	3.191e+02	3.091e+02	3.189e+02	2.425e+03	**3.051e+02**	8.277e+02	1.507e+03
	ET	3.911e+01	3.855e+01	3.865e+01	3.795e+01	3.809e+01	3.941e+01	3.778e+01	3.873e+01	3.875e+01	3.893e+01
F_{12}	**Best**	3.273e+03	3.401e+03	3.160e+03	2.900e+03	3.502e+03	3.485e+03	3.026e+03	3.728e+03	3.351e+03	3.191e+03
	Worst	4.180e+03	4.755e+03	4.471e+03	2.900e+03	5.242e+03	5.519e+03	4.665e+03	4.015e+03	5.705e+03	4.254e+03
	Mean	3.795e+03	4.139e+03	3.648e+03	2.900e+03	4.326e+03	4.247e+03	3.546e+03	3.838e+03	5.248e+03	3.695e+03
	STD	1.142e+03	1.496e+03	1.025e+03	**2.034e+02**	1.715e+03	1.645e+03	9.289e+02	1.160e+03	2.715e+03	1.049e+03
	ET	4.047e+01	3.979e+01	4.050e+01	4.044e+01	4.051e+01	3.976e+01	3.985e+01	4.058e+01	4.118e+01	3.997e+01
Mean Rank		5.6667	5.0000	5.5833	5.3333	8.0833	5.1667	6.2500	**4.0000**	4.5833	5.3333

well-documented limitations. This study confirms the importance of exploring alternative initialization methods.

Our findings demonstrated the effectiveness of various initialization schemes, including probabilistic distributions and low-discrepancy sequences, in influencing the performance of the AHO algorithm. Specifically, the experimental results on the CEC 2022 benchmark functions revealed that AHO's sensitivity to initialization schemes varies depending on the problem at hand. Some functions were

Table 5
Friedman test rankings of execution time for different dimensions

Dimension	D_0	D_1	D_2	D_3	D_4	D_5	D_6	S_1	S_2	S_3
$D = 10$	6.5000	6.5000	5.4167	4.0833	4.1667	5.9167	5.7500	5.2500	5.0833	6.3333
$D = 20$	6.1667	5.1667	6.4167	5.2500	4.3333	4.8333	4.6667	6.8333	6.0833	5.2500

more responsive to the choice of initialization, while others were relatively insensitive. Notably, the results showed that low-discrepancy sequences were particularly effective and exhibited greater stability, especially in higher dimensions. They consistently achieved the best results across most functions, underscoring their suitability as a robust initialization strategy, particularly in more complex and higher-dimensional optimization problems. This suggested that problem-specific characteristics, along with dimensionality, play a key role in determining the most appropriate initialization strategy.

Moving forward, we intend to conduct an in-depth study on the parameter initialization of MAs more broadly. In particular, we aim to explore innovative methods for automatic parameter initialization, which could further improve the adaptability and performance of these algorithms across a wide range of optimization problems. By incorporating novel approaches, such as hybrid techniques and dynamic parameter tuning, we seek to develop frameworks that allow metaheuristics to adjust their key parameters in response to the problem landscape. Additionally, we will focus on the exploration of hyper-parameter optimization strategies to systematically identify the most suitable configurations for enhancing both convergence speed and solution quality.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] S. J. Russell, P. Norvig, Artificial intelligence: a modern approach, Pearson, 2016.

[2] C. H. Papadimitriou, K. Steiglitz, Combinatorial optimization: algorithms and complexity, Courier Corporation, 2013.

[3] C. M. Bishop, Pattern recognition and machine learning, Springer google schola 2 (2006) 1122–1128.

[4] F. S. Hillier, G. J. Lieberman, Introduction to operations research, McGraw-Hill, 2015.

[5] D. Bertsimas, J. N. Tsitsiklis, Introduction to linear optimization, volume 6, Athena Scientific Belmont, MA, 1997.

[6] S. J. Wright, Numerical optimization, 2006.

[7] S. Boyd, L. Vandenberghe, Convex optimization, Cambridge university press, 2004.

[8] S. J. Wright, Primal-dual interior-point methods, SIAM, 1997.

[9] J. H. Holland, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, MIT press, 1992.

[10] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-international conference on neural networks, volume 4, ieee, 1995, pp. 1942–1948.

[11] A. Colorni, M. Dorigo, V. Maniezzo, et al., Distributed optimization by ant colonies, in: Proceedings of the first European conference on artificial life, volume 142, Paris, France, 1991, pp. 134–142.

[12] F. Zitouni, S. Harous, R. Maamri, The solar system algorithm: a novel metaheuristic method for global optimization, IEEE Access 9 (2020) 4542–4565.

[13] E. Cantú-Paz, On random numbers and the performance of genetic algorithms, Science Direct Working Paper No S1574-034X (04) (2002) 70205–7.

[14] Q. Li, S.-Y. Liu, X.-S. Yang, Influence of initialization on the performance of metaheuristic optimizers, Applied Soft Computing 91 (2020) 106193.

[15] N. Q. Uy, N. X. Hoai, R. I. McKay, P. M. Tuan, Initialising pso with randomised low-discrepancy

sequences: the comparative results, in: 2007 IEEE congress on evolutionary computation, IEEE, 2007, pp. 1985–1992.

[16] A. H. Gandomi, X.-S. Yang, S. Talatahari, A. H. Alavi, Firefly algorithm with chaos, Communications in Nonlinear Science and Numerical Simulation 18 (2013) 89–98.

[17] M. Lozano, A. Duarte, F. Gortázar, R. Martí, A hybrid metaheuristic for the cyclic antibandwidth problem, Knowledge-Based Systems 54 (2013) 103–113.

[18] F. Zitouni, S. Harous, A. Belkeram, L. E. B. Hammou, The archerfish hunting optimizer: A novel metaheuristic algorithm for global optimization, Arabian Journal for Science and Engineering 47 (2022) 2513–2553.

[19] R. Brits, A. P. Engelbrecht, F. van den Bergh, A niching particle swarm optimizer, in: Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning, volume 2, 2002, pp. 692–696.

[20] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization 11 (1997) 341–359.

[21] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360), IEEE, 1998, pp. 69–73.

[22] X.-S. Yang, S. Deb, Cuckoo search via lévy flights, in: 2009 World congress on nature & biologically inspired computing (NaBIC), Ieee, 2009, pp. 210–214.

[23] D. Karaboga, B. Basturk, Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems, in: International fuzzy systems association world congress, Springer, 2007, pp. 789–798.

[24] J. O. Agushaka, A. E. Ezugwu, L. Abualigah, S. K. Alharbi, H. A. E.-W. Khalifa, Efficient initialization methods for population-based metaheuristic algorithms: A comparative study, Archives of Computational Methods in Engineering 30 (2023) 1727–1787.

[25] X.-S. Yang, Nature-inspired metaheuristic algorithms, Luniver press, 2010.

[26] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, Advances in engineering software 69 (2014) 46–61.

[27] S. Mirjalili, A. Lewis, The whale optimization algorithm, Advances in engineering software 95 (2016) 51–67.

[28] A. P. Engelbrecht, Fundamentals of computational swarm intelligence, John Wiley & Sons, Inc., 2006.

[29] J. O. Agushaka, A. E. Ezugwu, Initialisation approaches for population-based metaheuristic algorithms: a comprehensive review, Applied Sciences 12 (2022) 896.

[30] S. Arora, S. Singh, Butterfly optimization algorithm: a novel approach for global optimization, Soft computing 23 (2019) 715–734.

[31] X. Wang, F. J. Hickernell, Randomized halton sequences, Mathematical and Computer Modelling 32 (2000) 887–899.

[32] E. I. Atanassov, Efficient cpu-specific algorithm for generating the generalized faure sequences, in: International Conference on Large-Scale Scientific Computing, Springer, 2003, pp. 121–127.

[33] P. Bratley, B. L. Fox, Algorithm 659: Implementing sobol's quasirandom sequence generator, ACM Transactions on Mathematical Software (TOMS) 14 (1988) 88–100.

[34] J. E. Gentle, Random number generation and Monte Carlo methods, volume 381, Springer, 2003.