MDPI

Article

Refining the Eel and Grouper Optimizer with Intelligent Modifications for Global Optimization

Glykeria Kyrou[1], Vasileios Charilogis[2] and Ioannis G. Tsoulos[3,*]

[1] Department of Informatics and Telecommunications, University of Ioannina, 47150 Kostaki Artas, Greece; g.kyrou@uoi.gr
[2] Department of Informatics and Telecommunications, University of Ioannina, 47150 Kostaki Artas, Greece; v.charilog@uoi.gr
[3] Department of Informatics and Telecommunications, University of Ioannina, 47150 Kostaki Artas, Greece;itsoulos@uoi.gr
* Correspondence: itsoulos@uoi.gr

Abstract: Global optimization is used in many practical and scientific problems. For this reason, various computational techniques have been developed. Particularly important are the evolutionary techniques, which simulate natural phenomena with the aim of detecting the global minimum in complex problems. A new evolutionary method is the Eel and Grouper Optimization (EGO) algorithm, inspired by the symbiotic relationship and foraging strategy of eels and groupers in marine ecosystems. In the present work, a series of improvements are proposed that aim both at the efficiency of the algorithm to discover the total minimum of multidimensional functions and at the reduction of the required execution time through the effective reduction of the number of functional evaluations. These modifications include the incorporation of a stochastic termination technique as well as an improvement sampling technique. The proposed modifications have been tested on multidimensional functions available from the relevant literature and compared with other evolutionary methods.

Keywords: Global optimization; Metaheuristic algorithms; Stochastic methods; Evolutionary algorithms; Swarm algorithms; Termination strategies; Sampling techniques.

1. Introduction

The goal of global optimization method aims to discover the global minimum of a continuous multidimensional function, and it is defined as

$$x^* = \arg\min_{x \in S} f(x) \tag{1}$$

with S:

$$S = [a_1, b_1] \times [a_2, b_2] \times \ldots [a_n, b_n]$$

The function $f(x)$ is defined as $f : S \to R, S \subset R^n$ and the set S denotes the bounds of x. of In recent years many researchers have published important reviews on global optimization [1–3]. Global optimization is a technique of vital importance in many fields of science and applications, as it allows finding the optimal solution to problems with multiple local solutions. In mathematics [4–7], it is used to solve complex mathematical problems, in physics [8–10], it is used to analyze and improve models that describe natural phenomena, in chemistry [11–13], it analyzes and designs molecules and chemical diagnostic tools, and in medicine[16] it analyzes and designs therapeutic strategies and diagnostic tools .

The methods that aim to discover the global minimum has two main categories, deterministic [17–19] and stochastic [20–22]. In the first category, there are techniques aimed at identifying the total minimum with some certainty, such as interval methods [23,24] and

Citation: Kyrou, G.; Charilogis, V.; Tsoulos, I.G. Refining the Eel and Grouper Optimizer with Intelligent Modifications for Global Optimization. *Journal Not Specified* **2024**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

are usually distinguished by their complex implementation. The vast majority of global optimization algorithms belong to stochastic methods that have simpler implementation and can also be applied to large - scale problems. Recently, Sergeyev et al [25] published a systematic comparison between deterministic and stochastic methods for global optimization problems.

An important branch of the stochastic methods are the evolutionary methods, that attempt to mimic a series of natural processes. Among these methods one can find the Differential Evolution method [26,27], Particle Swarm Optimization (PSO) methods [28–30], Ant Colony optimization methods [31,32], Genetic algorithms [33,34], the Exponential Distribution Optimizer [35], the Brain Storm Optimization method [36] etc. Additionally, since in recent years there has been an extremely wide spread of parallel computing units, many researches have proposed evolutionary methods that exploit modern parallel processing units [37–39].

Among the evolutionary techniques one finds a large group of methods that have been explored intensively in recent years, the so - called Swarm Intelligence algorithms. These methods [40–42] are inspired by the collective behavior of swarm. These algorithms mimic systems in which candidate solutions interact locally and cooperate worldwide to discover the global minimum of any problem. These algorithms are very important tools for dealing with complex optimization problems in many applications [43].

In addition to the previously mentioned Particle Swarm Optimization and Ant Colony methods techniques that are also included in Swarm intelligence algorithms, other methods that belong to this category are, the Fast Bacterial Swarming Algorithm (FBSA) [44], the Fish Swarm Algorithm [45], the Dolphin Swarm Algorithm [46], the Whale Optimization Algorithm (WOA) algorithm 47–50, the Tunicate Swarm Algorithm [51], the Sine Cosine algorithm (SCA) algorithm [52–55], the Salp Swarm algorithm (SSA) algorithm [56–59], the Exponential Distribution Optimizer (EDO) [35], Brain Storm Optimization, etc. These methods simulate a series of complex interactions between biological species [60,61], such as:

1. Naturalism: Where two species can live without affecting each other.
2. Predation, where one creature dies by feeding another.
3. Parasitism: where one species can cause harm to another.
4. In competitive mode, the same or different organizations compete for resources.
5. Mutualism [62–64]: when two organisms have a beneficial interaction.

Among swarm intelligence algorithms one can find the Eel and Grouper (EGO) algorithm, which is inspired by the symbiotic interaction and foraging strategy of eels and groupers in marine ecosystems. Bshary et al. [65] consider that target ingestion, something observed in eels and groupers, is a necessary condition for interspecific cooperative hunting to occur. Intraspecific predation could increase the hunting efficiency of predators by mammals. According to Ali Mohammadzadeh and Seyedali Mirjalili the EGO optimization algorithm [66] generates a set of random answers, then stores the best answers found so far, allocates them to the target point, and changes the answers with them. As the number of iterations increases, the limits of the sine function are changed to enhance the phase of finding the best solution. This method stops the process when the iteration exceeds the maximum number. Because the EGO optimization algorithm generates and boosts a collection of random responses, it has the advantage of increased local optimum discovery and avoidance compared to individual methods. According to Ali Mohammadzadeh and Seyedali Mirjalili, the algorithm's capabilities extend to NP-hard problems in wireless sensor networks[67], IoT[68], logistics[69], smart agriculture[70], bioinformatics[71] and machine learning[72] in various fields such as programming, image segmentation[73], electrical circuit design[74], feature selection and 3D path planning in robotics[75].

This paper introduces some modifications to the EGO algorithm in order to improve its efficiency. The proposed amendments are presented below:

- The addition of a sampling technique based on the K-means method [76,77,79]. The sampling points will facilitate finding the global minimum of the function in the most

efficient way. Additionally, by applying this method, nearby points are discarded. Initialization of the population of evolutionary techniques is a crucial factor which may accelarate the**.** The initialization of populations in evolutionary techniques can push these techniques to more efficiently locate the global minimum, and in this direction a multitude of research works have been presented in recent years, such as the work of Maaranen et al [80], where they apply quasi-random sequences in the initial population of a genetic algorithm. Likewise, Paul et al [81] suggested a method for the initialization of the population of genetic algorithms using a Vari-begin and Vari-diversity (VV) seeding method. Ali et al. proposed a series of initialization methods for the Differential Evolution method [82]. A novel method that initializes the population of evolutionary algorithms using clustering and Cauchy deviates is suggested in the work of Bajer et al [83]. A systematic review of initialization techniques for evolutionary algorithms can be found in the work of Kazimipour et al [84].

- Using a termination technique that is developed with random measurements. Each time the algorithm is repeated, the minimum value is recorded. When this remains constant for a pre - defined number of iterations, the process is terminated. Therefore, the method will terminated without wasting execution time in iterations, avoiding unnecessary consumption of computing resources. There are several methods found in the recent bibliography to terminate optimization methods. An overview of methods used to terminate evolutionary algorithms can be found in the work of Jain et al [86]. Also, Zielinski et al outlined some stopping rules used particularly in the Differential Evolution method [87]. Recently, Ghoreishi et al. published a literature study concerning various termination criteria on evolutionary algorithms [88]. Moreover, Ravber et al. performed an extended research on the impact of maximum number of iterations to the effectiveness of evolutionary algorithms [89].
- Application of randomness in the definition of the range of the positions of candidate solutions.

The rest of this paper is divided into the following sections: in section 2, the proposed method is fully described, in section 3 the experimental results and statistical comparisons are outlined and finally in section 4 some conclusions and guidelines for future improvements are discussed.

2. The proposed method

The main steps of the proposed algorithm are discussed in this section. Also, the mentioned modifications are fully described.

2.1. The main steps of the algorithm

EGO optimization algorithm starts by initializing a population consisting of "search agents" that search to find the optimal solution. At each iteration, the position of the "prey" (optimal solution) is calculated. Agent positions are adjusted based on random variables and their distance based on the optimal position.At the end of each iteration, the current solutions are compared and it is decided whether the algorithm should continue or terminate. The steps of the proposed method are provided in Algorithm 1. Also, the algorithm is presented as a series of steps in the flowchart of Figure 1. Using flowchart and algorithm simultaneously enhances visual perception and detailed analysis of logic and processes.

Algorithm 1 EGO Algorithm

Initialization step.

1. **Define** as N_c the number of elements in the search Agents
2. **Define** as N_g, the maximum number of allowed iterations.
3. **Initialize** randomly the search agents $x_i,\ i = 1, \ldots, N_c$ in set S.
4. **Set** $t = 0$, the iteration counter.
5. **Set** $s_r = 0$, the starvation rate of the algorithm.
6. **Set** $m = 1$, this parameter influences how the variables f_1, f_2 are defined, which in turn affects the calculation of new positions. When $m = 2$, it introduces randomness to the range of positions before the update, while in the inactive state, the range remains fixed.

Calculation step.

1. Update variables a and s_r:
   - $a = 2 - 2\frac{t}{N_g}$
   - $s_r = 100\frac{t}{N_g}$
2. Compute the fitness of each search agents.
3. Sort all solutions according to their fitness values.
4. Set XP the estimated position of the prey.
5. **For** $j = 1, \ldots, N_c$ **do**
   (a) **Update** random variables $r_1, r_2, r_3, r_4, C_1, C_2, b$ **:**
      - $r_1$and r_2 are random numbers in $[0, 1]$,
      - $r_3 = (a - 2)r_1 + 2$
      - $r_4 = 100r_2$
      - $C_1 = 2 * a * r_1 - a$
      - $C_2 = 2 * r_1$
      - $b = a * r_2$

   (b) $X_1 = e^{br_3} \sin(2\pi r_3) C_1 |\mathbf{XE} - \mathbf{XP}| + \mathbf{XE}$, **where** $\mathbf{XE} = C_2 x_j$

   (c) $X_2 = x_j + C_1 \left| x_j - \mathbf{XP} \right|$

   (d) **if** $(r4 \leq s_r)$
      - **if** $m = 1$ **then set** $f_1 = 0.8, f_2 = 0.2$
      - **else set** f_1 to a random number in $[0, 2]$ and f_2 to a random number in $[-2, 2]$

   (e) **Update** agent: $x_j = \frac{f_1 X_1 + f_2 X_2}{2}$
6. **End For**

Termination check step

1. **Set** $t = t + 1$
2. **If** $t \geq N_g$ **terminate.**
3. Calculate the stopping that proposed in the work of Charilogis [78]. In the Similarity stopping rule, at every iteration t, the absolute difference between the current located global minimum $f_{min}^{(t)}$ and the previous best value $f_{min}^{(t-1)}$ is calculated:

$$\delta^t = \left| f_{\min}^{(t)} - f_{\min}^{(t-1)} \right| \tag{2}$$

The algorithm terminated when $\delta^t \leq \epsilon$ for N_k consecutive iterations, where ϵ is a small positive value. If the termination criteria are not satisfied then go to Calculation step, else terminate and return the best solution.

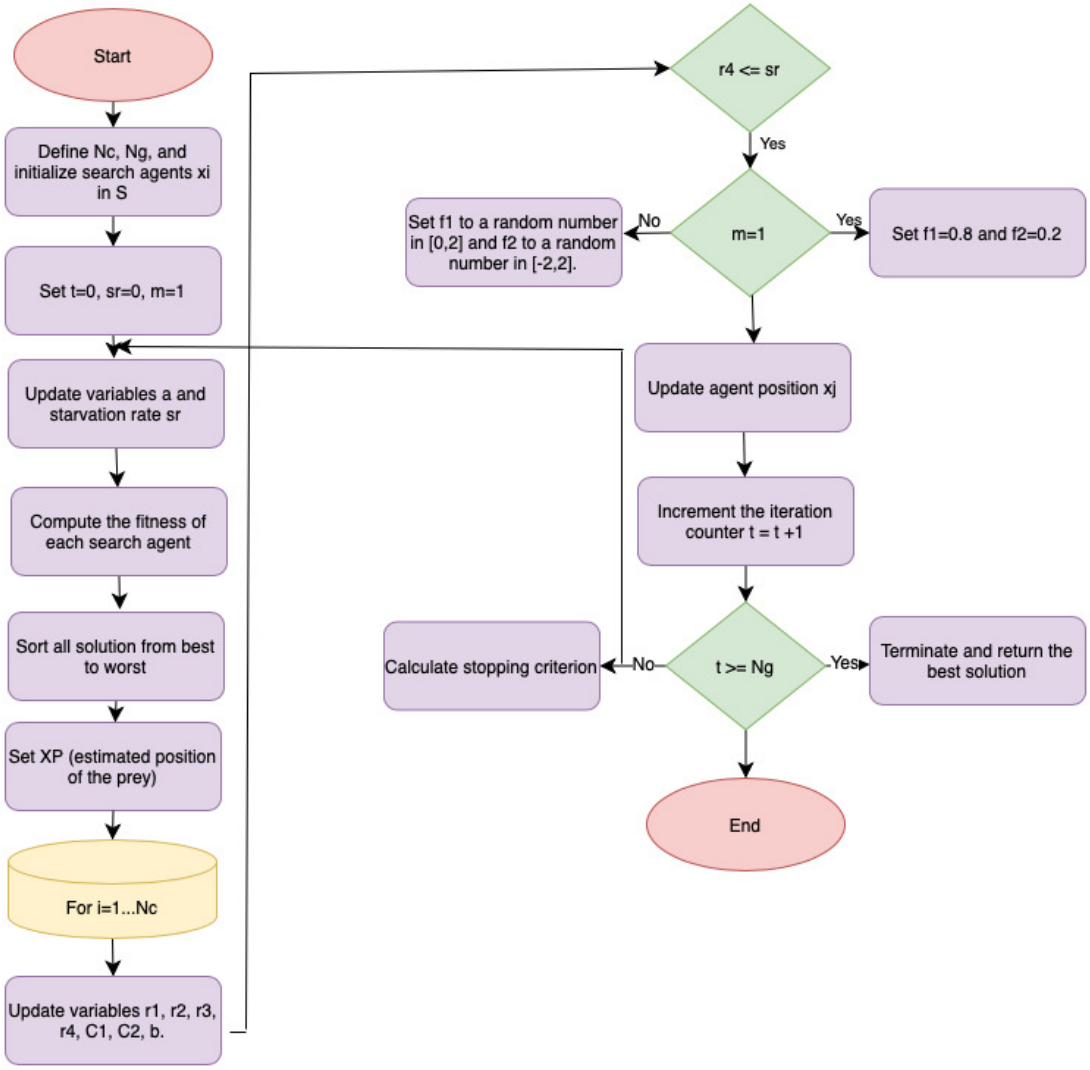

Figure 1. Flowchart of the suggested global optimization procedure.

The main modifications introduced by the proposed algorithm are the following:

1. The members of the population are initialized using a procedure that incorporates the K-means algorithm. This procedure is fully described in subsection 2.2. The purpose of this sampling technique is to produce points that are close to the local minima of the objective problem through systematic clustering. This will potentially significantly reduce the time required to complete the technique. The samples used in the proposed algorithm are the calculated centers of the K-means algorithm.
2. The second modification proposed is the stopping rule invoked at every step of the algorithm. This rule measures the similarity between the best fitness values obtained between consecutive iterations of the algorithm. If this difference takes low values for a consecutive number of iterations, then the algorithm may not be able to find a lower value for the global minimum and should stop.
3. The third modification is the m flag, which controls the randomness in the range of candidate solutions. When this value is set to 2, then the critical parameters f_1, f_2 are calculated using random numbers.

2.2. The used sampling procedure

The used sampling procedure that was incorporated in this work initially generates samples from the objective problem. Then, using the K-means method, only the estimated

centers are selected as samples for the proposed algorithm. This technique, which is an achievement of James MacQueen [79], is one of the most well-known clustering algorithms in the broad research community, both in data analysis and in machine learning [85] and pattern recognition [90]. The algorithm aims to divide a data set into k clusters. The K-means algorithm tries to divide the data into groups in such a way that the internal points of each group are as close as possible to each other. At the same time, he tries to place the central points of each group in positions which are as representative as possible for the points of their group. During the past years a series of variants of this algorithm has been proposed, such as the Genetic K-means algotithm [91], the unsupervised K-means algorithm [92], the Fixed-centered K-means algorithm [93] etc. A review of K-Means clustering algorithms can be found in the work of Oti et al. [94] Next, the basic steps of the algorithm are provided in Algorithm 2. A flowchart of the K-Means procedure is also depicted in Figure 2.

Algorithm 2 K-means Algorithm

1. **Initialization**
   (a) **Set** k the number of clusters.
   (b) **Obtain** randomly the initial samples $x_i,\ i = 1, \ldots, N_m$
   (c) **Set** $S =_j \{\ \}$, from $j = 1, \ldots, k$.
2. Repeat
   (a) **For** every point $x_i,\ i = 1, ..., N_m$ **do**
      i. **Set** $j^* = \text{argmin}_{m=1}^{k}\{D(x_i, c_m)\}$. j^* is the nearest center from x_i
      ii. **Set** $S_{j^*} = S_{j^*} \cup \{x_i\}$.
      iii. **End For**
   (b) **For** every center $c_j,\ j = 1..k$ **do**
      i. **Set** M_j the number of samples in S_j
      ii. **Update** the center c_j as

$$c_j = \frac{1}{M_j} \sum_{x_i \in S_j} x_i$$

   (c) **End** For
3. Terminate when c_j no longer changes
4. The final samples of the algorithm are the centers c_j.

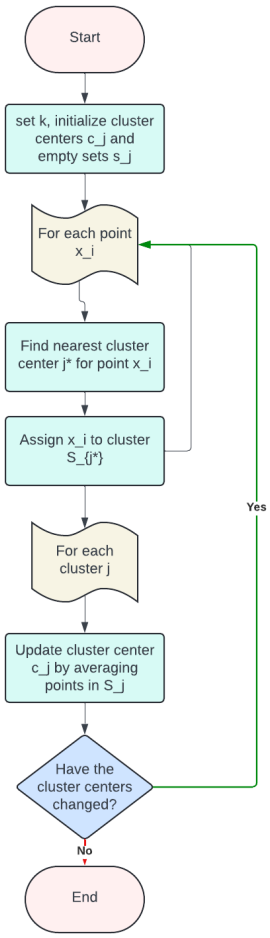

Figure 2. The flowchart of the K-Means procedure.

3. Results

This section will begin with a detailed description of the functions that will be used in the experiments, followed by an analysis of the experiments performed and comparisons with other global optimization techniques.

3.1. Test functions

The test functions used in the experiments have been suggested in a series of relative works [95,96] and they originated in a series of scientific fields. Also, these objective functions have been studied in various publications [97–101]. Also, a series of function founded in [102] are used as test functions. The used functions are defined as follows:

- **Ackley** function:

$$f(x) = -a \exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(cx_i)\right) + a + \exp(1)$$

  with a=20.0.

- **Bf1** (Bohachevsky 1) function:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$$

- **Bf2** (Bohachevsky 2) function:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$$

- **Bf3** (Bohachevsky 3) function:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1 + 4\pi x_2) + \frac{3}{10}$$

- **Branin** function:

$$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$$

with $-5 \leq x_1 \leq 10,\ 0 \leq x_2 \leq 15$.

- **Camel** function:

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, \quad x \in [-5, 5]^2$$

- **Easom** function:

$$f(x) = -\cos(x_1)\cos(x_2)\exp\left((x_2 - \pi)^2 - (x_1 - \pi)^2\right)$$

with $x \in [-100, 100]^2$.

- **Equal maxima** function, defined as:

$$f(x) = \sin^6(5\pi x)$$

- **Exponential** function, with the following definition:

$$f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right), \quad -1 \leq x_i \leq 1$$

The values $n = 4, 8, 16, 32$ were used in the conducted experiments.

- **F9** test function:

$$f(x) = -\sum_{i=1}^{n}(10 + 9\cos(2\pi k_i x_i))$$

with $x \in [0, 1]^n$.

- **Extended F10** function:

$$f(x) = \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i)\right)$$

- **F14** function:

$$f(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$$

- **F15** function:

$$f(x) = \sum_{i=1}^{11}\left(a_i - \frac{x_1(b_i + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right)^2$$

- **F17** function:

$$f(x) = \left(1 + (x_1 + x_2 + 1)^2 \times \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right) \times \left(30 + (2x_1 - 3x_2)^2 \times \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right)$$

- **Five - uneven - peak trap** function:

$$f(x) = \begin{cases} 80(2.5 - x) & 0 \leq x < 2.5 \\ 64(x - 2.5) & 2.5 \leq x < 5.0 \\ 64(7.5 - x) & 5.0 \leq x < 7.5 \\ 28(x - 7.5) & 7.5 \leq x < 12.5 \\ 28(17.5 - x) & 12.5 \leq x < 17.5 \\ 32(x - 17.5) & 17.5 \leq x < 22.5 \\ 32(27.5 - x) & 22.5 \leq x < 27.5 \\ 80(x - 27.5) & 27.5 \leq x \leq 30 \end{cases}$$

- **Himmelblau** function:

$$f(x) = 200 - \left(x_1^2 + x_2 - 11\right)^2 - \left(x_1 + x_2^2 - 7\right)^2$$

with $x \in [-6, 6]^2$.

- **Griewank2** function:

$$f(x) = 1 + \frac{1}{200} \sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2} \frac{\cos(x_i)}{\sqrt{(i)}}, \quad x \in [-100, 100]^2$$

- **Griewank10** function. The function is given by the equation

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

with $n = 10$.

- **Gkls** function [103]. The function $f(x) = \text{Gkls}(x, n, w)$, is a test function proposed in [103] with w local minima. The values values $n = 2, 3$ and $w = 50$ were used in the conducted experiments.
- **Goldstein and Price function**

$$\begin{aligned} f(x) \quad = \quad & \Big[1 + (x_1 + x_2 + 1)^2 \\ & \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\Big] \times \\ & \left[30 + (2x_1 - 3x_2)^2\right. \\ & \left.\left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right] \end{aligned}$$

With $x \in [-2, 2]^2$.

- **Hansen** function: $f(x) = \sum_{i=1}^{5} i \cos[(i-1)x_1 + i] \sum_{j=1}^{5} j \cos[(j+1)x_2 + j]$, $x \in [-10, 10]^2$.
- **Hartman 3** function:

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$$

$$\text{with } x \in [0,1]^3 \text{ and } a = \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}, \; c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix} \text{ and } \quad (204)$$

$$p = \begin{pmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{pmatrix}$$

- **Hartman 6** function: (205)

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$$

$$\text{with } x \in [0,1]^6 \text{ and } a = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}, \; c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix} \text{ and } \quad (206)$$

$$p = \begin{pmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{pmatrix}$$

- **Potential** function, this function represents the energy of a molecular conformation of N atoms. The interaction of these atoms is determined by the Lennard-Jones potential [104]. The definition of this potential is: (207, 208, 209)

$$V_{LJ}(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

For the conducted experiments the values $N = 3, 5$ were used. (210)

- **Rastrigin** function. (211)

$$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2), \quad x \in [-1,1]^2$$

- ***Rosenbrock*** *function*. (212)

$$f(x) = \sum_{i=1}^{n-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right), \quad -30 \leq x_i \leq 30. \quad (213)$$

The values $n = 4, 8, 16$ were incorporated in the conducted experiments. (214)

- **Shekel 5** function. (215)

$$f(x) = -\sum_{i=1}^{5} \frac{1}{(x - a_i)(x - a_i)^T + c_i}$$

$$\text{with } x \in [0,10]^4 \text{ and } a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \end{pmatrix}, \; c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \end{pmatrix} \quad (216)$$

- **Shekel 7** function. (217)

$$f(x) = -\sum_{i=1}^{7} \frac{1}{(x - a_i)(x - a_i)^T + c_i}$$

with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 3 & 5 & 3 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \end{pmatrix}$.

- **Shekel 10** function.

$$f(x) = -\sum_{i=1}^{10} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.6 \end{pmatrix}$.

- **Sinusoidal** function defined as:

$$f(x) = -\left(2.5\prod_{i=1}^{n}\sin(x_i - z) + \prod_{i=1}^{n}\sin(5(x_i - z))\right), \quad 0 \leq x_i \leq \pi.$$

The values $n = 4, 8, 16$ were incorporated in the conducted experiments.

- **Schaffer** function:

$$f(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$$

- **Schwefel221** function:

$$f(x) = 418.9829n + \sum_{i=1}^{n} -x_i \sin\left(\sqrt{|x_i|}\right)$$

- **Schwefel222** function:

$$f(x) = \sum_{i=1}^{n}|x_i| + \prod_{i=1}^{n}|x_i|$$

- **Shubert** function:

$$f(x) = -\prod_{i=1}^{n}\sum_{j=1}^{5} j\cos((j+1)x_i + j)$$

with $x \in [-10,10]^n$

- **Sphere** function:

$$f(x) = \sum_{i=1}^{n} x_i^2$$

- **Test2N** function:

$$f(x) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i, \quad x_i \in [-5,5].$$

The values $n = 4, 5, 6, 7$ were incorporated for the conducted experiments.

- **Test30N** function:

$$f(x) = \frac{1}{10}\sin^2(3\pi x_1)\sum_{i=2}^{n-1}\left((x_i-1)^2\left(1+\sin^2(3\pi x_{i+1})\right)\right)+(x_n-1)^2\left(1+\sin^2(2\pi x_n)\right)$$

For the conducted experiments the values $n = 3, 4$ were used.

- **Uneven decreasing maxima** function:

$$f(x) = \exp\left(-2\log(2)\left(\frac{x-0.08}{0.854}\right)^2\right)\sin^6\left(5\pi\left(x^{\frac{3}{4}}-0.05\right)\right)$$

- **Vincent** function:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n}\sin(10\log(x_i))$$

with $x \in [0.25, 10]^n$.

3.2. Experimental results

A series of global optimization methods were applied to the mentioned test functions. All experiments were performed 30 times using different seed for the random generator and the average value of function calls was measured. The used software was coded in ANSI C++ using the freely available OPTIMUS optimization environment, that is available from https://github.com/itsoulos/OPTIMUS (accessed on 27 September 2024). The experiments were executed on a Debian Linux system that runs on an AMD Ryzen 5950X processor, with 128GB of RAM. In all cases the BFGS [105] local optimization method was used at the end of each global optimization technique to ensure that an actual minimum will be discovered by the global optimization method. The values for all experimental parameters are shown in Table 1.

Table 1. The values used for every parameter of the used algorithms.

PARAMETER	MEANING	VALUE
N_c	Number of chromosomes/particles	200
N_g	Maximum number of allowed iterations	200
N_m	Number of samples for the K-means	$10 \times N_c$
N_k	Number of maximum iterations for stopping rule	5
p_s	Selection rate of the genetic algorithm	0.1
p_m	Mutation rate of the genetic algorithm	0.05

The experimental results for the test functions and a series of optimization methods are shown in Table 2, where the following applies to this table:

- The column FUNCTION represents the used test function.
- The column GENETIC represents the usage of a genetic algorithm [33,34] to the test function. This genetic algorithm is equipped with N_c chromosomes and the maximum number of generations is set to N_g. The modified version of Tsoulos [106] was used as a genetic algorithm here.

- The column PSO represents the incorporation of a Particle Swarm Optimizer [29,30] to each test function. This algorithm has N_c particles and the maximum number of allowed iterations is set to N_g. For the conducted experiments, the improved PSO method as proposed by Charilogis and Tsoulos was used [107].
- The column DE refers to the Differential Evolution method [26,27].
- The column EGO represents the initial method without the modifications suggested in this work.
- The column EEGO represents the usage of the proposed method. The corresponding settings are shown in Table 1.
- The row SUM is used to measure the total function calls for all problems.

Table 2. Experimental results using the incorporated optimization methods. Numbers in cells stand for average function calls.

FUNCTION	GENETIC	PSO	DE	EGO	EEGO
ACKLEY	6749	6885	10220	8714	4199
BF1	4007	4142	8268	4762	3228
BF2	3794	3752	7913	4299	2815
BF3	3480	3306	10270	3747	2501
BRANIN	2376	2548	4101	2659	1684
CAMEL	2869	2933	5609	3317	2262
EASOM	1958	1982	2978	2235	1334
EQUAL_MAXIMA	2651	1499	2374	2013	1286
EXP4	2946	3404	5166	3392	2166
EXP8	3120	3585	5895	3347	2802
EXP16	3250	3735	6498	3345	3279
EXP32	3561	3902	7606	3332	3430
EXTENDEDF10	4862	3653	5728(0.87)	4737	2609
FIVE_UNEVEN	3412(0.67)	3913(0.87)	4042(0.14)	5006(0.97)	3849(0.90)
F9	2604	1888	2271	2748	1439
F14	6686	5498	5279(0.63)	9228(0.94)	6063
F15	4373	6696	5874(0.80)	7342	4397
F17	3667	3805	10441	4057	2766
HIMMELBLAU	2481	1013	6636	1718	1119
GKLS250	2280	2411	3834	3332	1603
GKLS350	2613	2234	3919	2493	1298
GOLDSTEIN	3687	3865	6781	4015	2784
GRIEWANK2	4501	3076 (0.73)	7429	4682	2589 (0.96)
GRIEWANK10	6410 (0.97)	8006	18490	8772	7435
HANSEN	3210	2856	4185	3789	2484
HARTMAN3	2752	3140	5190	3078	1793
HARTMAN6	3219	3710	5968	3583	2478
POTENTIAL3	4352	4865	6118	6027	4081
POTENTIAL5	7705	9183	9119	9968	8886
RASTRIGIN	4107	3477	6216	4201	2304
ROSENBROCK4	3679	6372	8452	6137	4019
ROSENBROCK8	5270	8284	11530	8569	6801
ROSENBROCK16	8509	11872	17432	11777	11996
SHEKEL5	3325	4259	6662	3948	2495
SHEKEL7	3360	4241	6967	4043	2432
SHEKEL10	3488	4237	6757	3932	2516
SHUBERT2	3567	2123	3526	3622	2300
SHUBERT4	3358	1823	3067	3593	1967(0.97)
SHUBERT8	3569	2348	3120(0.94)	2862	2267
SCHAFFER	18787	15176	6315	28679	23531
SCHWEFEL221	2667	2529	5415	3426	2203
SCHWEFEL222	33725	42898	12200	51654	38876
SPHERE	1588	1521	3503	1642	1162
TEST2N4	3331	3437	6396	3695	2277
TEST2N5	4000	3683	6271	4234	2734 (0.96)
TEST2N6	4312 (0.93)	3781	5410 (0.93)	4599	2905 (0.86)
TEST2N7	4775 (0.90)	4060	7074 (0.97)	5146	3559 (0.73)
SINU4	2991	3504	5953	3478	2005
SINU8	3442	4213	6973	4420	3158
SINU16	4320	5019	6979	7033	5891
TEST30N3	3211	4610	6168	3971	2362
TEST30N4	3679	4629	7006	4908	2978
UNEVEN_MAXIMA	2969	2729	2393	2972	1560
VINCENT2	12779	1797(0.87)	6216	2094	1834(0.90)
VINCENT4	19385	1830(0.67)	4691(0.83)	2674	2697(0.64)
VINCENT8	19882	2717	4417(0.77)	3423	4368(0.77)
SUM	**297650**	**268654**	**288413**	**320469**	**231856**

In figure 3 we present the total function calls of every optimization method is presented graphically. The proposed method has excellent results compared to the other optimization techniques according to the experiment we conducted. As we can observe it has the least number of calls than all the other techniques.

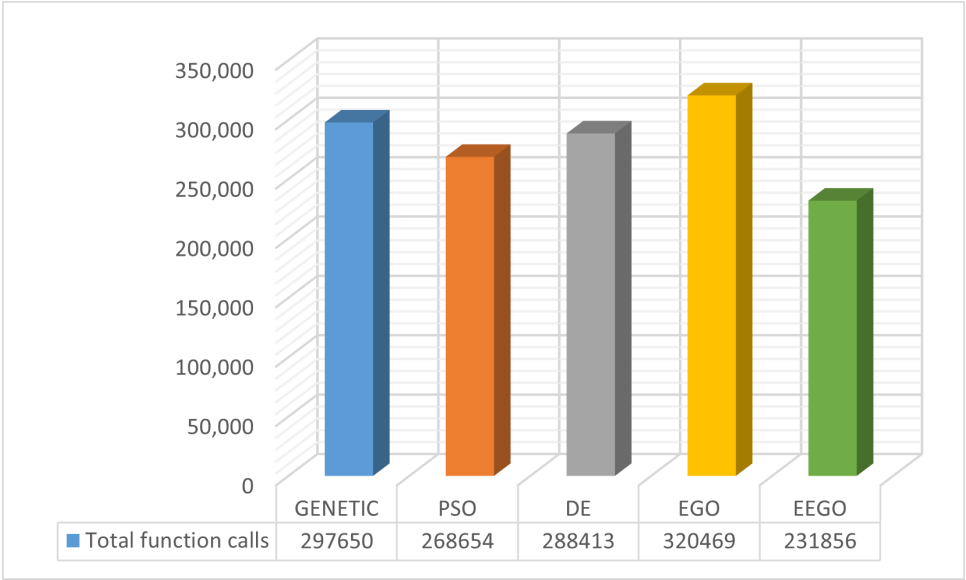

Figure 3. Total function calls for the incorporated optimization methods.

. Furthermore, as the experimental results clearly indicate, the modified version outperforms the original EGO method in terms of average and function calls and this is depicted in Figure 4 that outlines box plots for the mentioned methods.

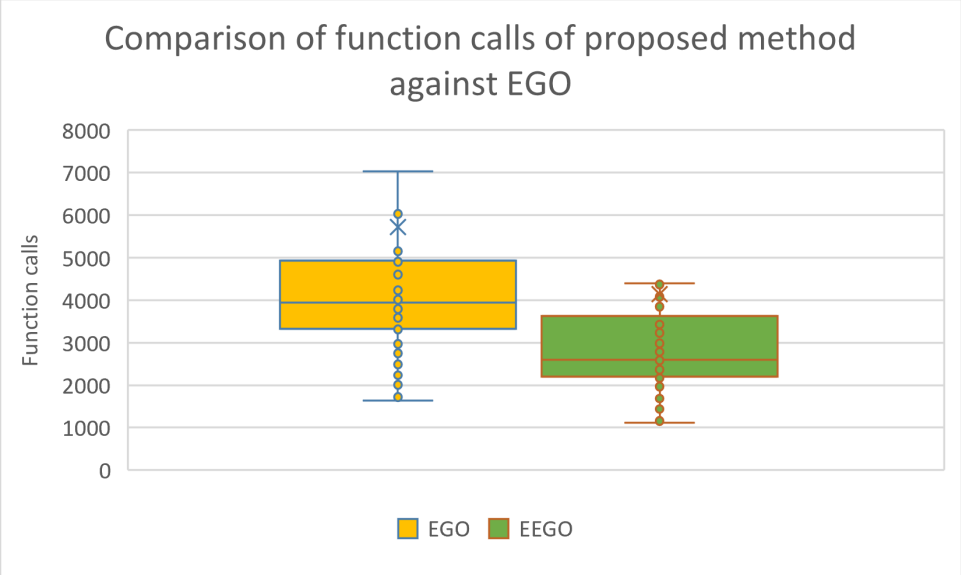

Figure 4. Box plot used to compare the methods EGO and the modified version as suggested in the current work.

A box plot between all the used methods is depicted in in Figure 5.

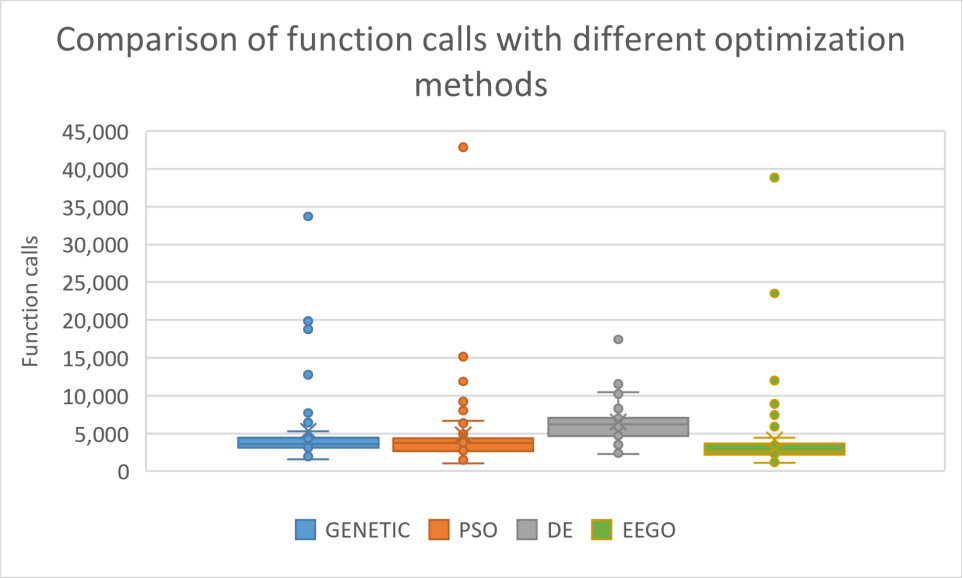

Figure 5. Comparison of average function calls for the incorporated optimization methods, using proposed initial distribution

Moreover, a statistical comparison for all used methods is outlined in Figure 6.

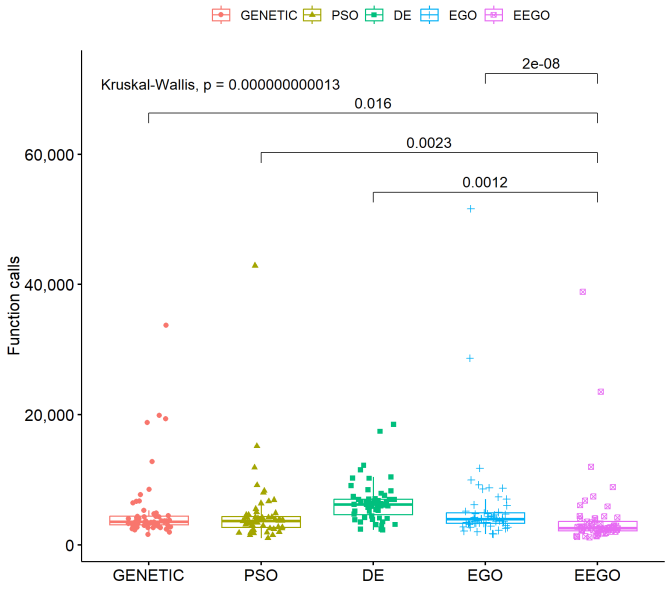

Figure 6. Statistical comparison of all used methods.

According to Table 2 and Figure 6, the EEGO method proves to be more efficient, as it consistently requires fewer function calls compared to the DE, PSO, GENETIC, and EGO methods, and these differences are statistically significant ($p < 0.05$). For example, in the ACKLEY function, EEGO required 4199 calls, while DE required 10.220, PSO 6885, and GENETIC 6749, with a p-value < 0.05, showing that EEGO is significantly more efficient. Similar examples are observed in the BF1 function, where EEGO had 3228 calls, while DE required 8268 ($p < 0.05$), and in the BRANIN function, where EEGO required 1684 calls, compared to 4101 for DE ($p < 0.05$). Overall, EEGO shows the best performance in most cases, with statistically significant differences in function calls, as it effectively reduces the number of calls compared to the other methods. While PSO and GENETIC perform better

than DE in some instances, they still lag significantly behind EEGO. The p-values confirm that these differences are statistically significant, indicating that EEGO is the most efficient method overall.

One more experiment which was performed with the ultimate goal of measuring the importance of K-means sampling in the proposed method. The results for this experiment are outlined in Table 3 and the following sampling methods were used:

1. The column UNIFORM represents the usage of uniform sampling in the current method.
2. The column TRIANGULAR stands for the usage of the triangular distribution [108] for sampling.
3. The column MAXWELL represents for the application of the Maxwell distribution [109] to produce initial samples for the used method.
4. The column KMEANS represents the usage of the method described in subsection 2.2 to produce initial samples for the used method.

Table 3. A series of sampling techniques is used in the proposed method. Numbers in cells denote average function calls.

FUNCTION	UNIFORM	TRIANGULAR	MAXWELL	KMEANS
ACKLEY	6118	5912	5986	4199
BF1	4513	4318	4055	3228
BF2	3959	3879	3587	2815
BF3	3506	3344	3129	2501
BRANIN	2282	2131	2066	1684
CAMEL	3156	2919	2848	2262
EASOM	1756	1650	1321	1334
EQUAL_MAXIMA	1445	1293	1165	1286
EXP4	3438	3273	3194	2166
EXP8	3432	3387	3152	2802
EXP16	3369	3326	3291	3279
EXP32	3216	3225	3344	3430
EXTENDEDF10	4304	3913	3718	2609
FIVE_UNEVEN	4685	4330	4358	3849
F9	1958	1681	1878	1439
F14	7552	7573	6148	6063
F15	6806	6466	6543	4397
F17	3805	3700	3543	2766
HIMMELBLAU	1333	1173	1114	1119
GKLS250	2268	2023	1778	1603
GKLS350	2151	1841	2069	1298
GOLDSTEIN	3855	3731	3530	2784
GRIEWANK2	4310	4510	4035	2589
GRIEWANK10	8640	8773	8232	7435
HANSEN	3329	3071	2734	2484
HARTMAN3	2849	2673	2678	1793
HARTMAN6	3456	3249	3119	2478
POTENTIAL3	4554	5095	3928	4081
POTENTIAL5	8356	10032	7504	8886
RASTRIGIN	3310	3187	2751	2304
ROSENBROCK4	6566	6353	5588	4019
ROSENBROCK8	8379	8717	7783	6801
ROSENBROCK16	11921	12471	11677	11996
SHEKEL5	3946	3731	3859	2495
SHEKEL7	3990	3646	3944	2432
SHEKEL10	3836	3630	3694	2516
SHUBERT2	3288	3212	2631	2300
SHUBERT4	3116	2919	2499	1967
SHUBERT8	2815	2810	2337	2267
SCHAFFER	55131	40715	60717	23531
SCHWEFEL221	2724	2901	2799	2203
SCHWEFEL222	53118	54593	55354	38876
SPHERE	1346	1188	1084	1162
TEST2N4	3345	3233	2867	2277
TEST2N5	3937	3742	3094	2734
TEST2N6	4008	4473	3266	2905
TEST2N7	4545	4612	3549	3559
SINU4	3128	2879	3559	2005
SINU8	4126	3767	5637	3158
SINU16	6774	5977	7739	5891
TEST30N3	3704	3384	3175	2362
TEST30N4	4262	4327	3491	2978
UNEVEN_MAXIMA	1877	1810	1666	1560
VINCENT2	1598	1482	1849	1834
VINCENT4	2471	2282	2296	2697
VINCENT8	3074	2797	2883	4368
SUM	**324736**	**307329**	**315835**	**231856**

Initial distributions play a critical role in a wide range of applications, including optimization, statistical analysis, and machine learning. Table 3 presents the proposed distribution alongside other established distributions. The uniform distribution is widely used due to its ability to evenly cover the search space, making it suitable for initializing optimization algorithms [110]. The triangular distribution is applied in scenarios where there is knowledge of the bounds and the most probable value of a phenomenon, making it useful in risk management models [111]. The Maxwell distribution, although originating from physics, finds applications in simulating communication networks, where data transfer speeds can be modeled as random variables [112]. Finally, the K-means method is used for data clustering, with K-means++ initialization offering improved performance compared to random distributions, particularly in high-dimensional problems [113]. As observed in Table 3, the choice of an appropriate initial distribution can significantly affect the performance of the algorithms that utilize them.

In the scatter plot in figure 7, the critical parameter "p" was found to be very small, leading to the rejection of the null hypothesis and indicating that the experimental results are highly significant. Scatter plot for different initial distributions.

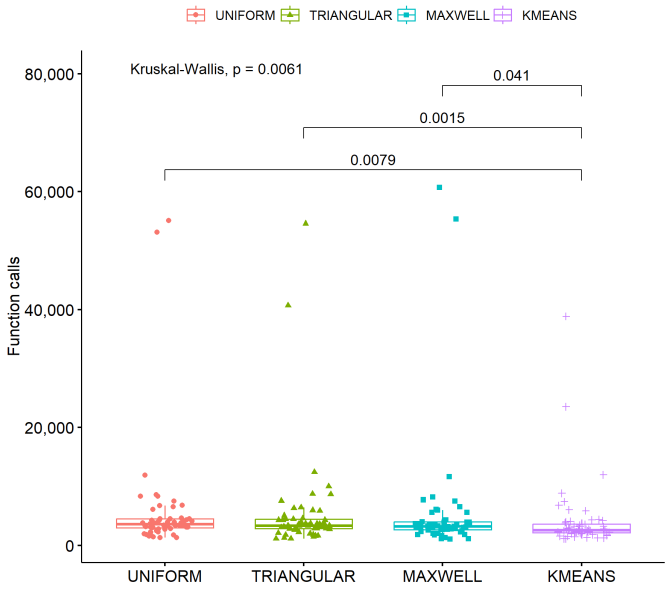

Figure 7. Scatter plot for different initial distributions

In addition, in order to investigate the impact that the choice of sampling method has on the optimization method, an additional experiment was done, in which the execution time of the proposed method was recorded and with different sampling techniques for the Rosenbrock function, in which function the dimension varied between 2 and 50. The results from this experiment are graphically outlined in Figure 8.

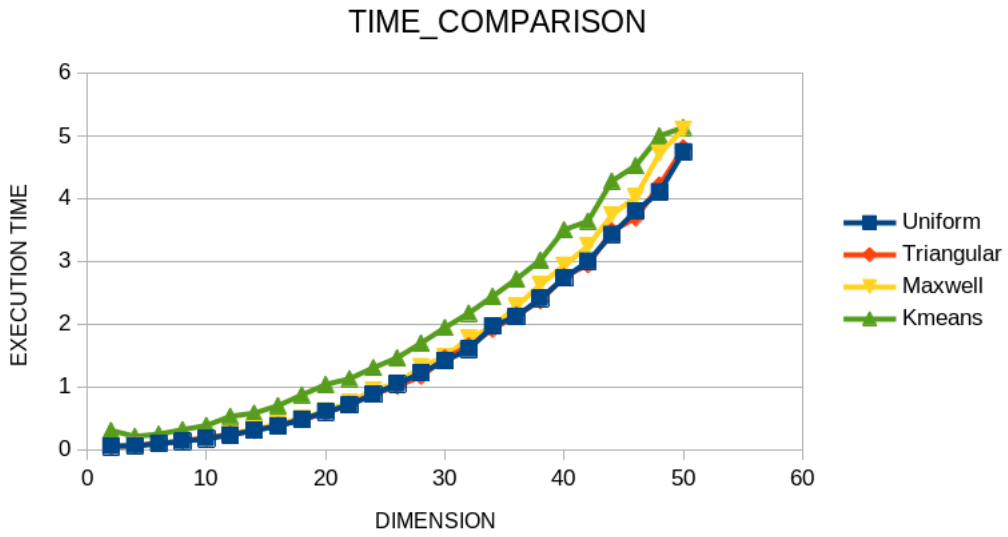

Figure 8. Time comparison for the Rosenbrock function and the proposed optimization method using the four sampling techniques mentioned before. The time depicted in figure is the sum of the execution times for 30 independent runs.

The proposed sampling method can significantly increase the required execution time, since an iterative process is required before starting the algorithm. The next sampling procedure in required execution time appears to be the Maxwell sampling, but not significantly compared to uniform sampling.

4. Conclusions

The article proposed some modifications to the EEGO optimization method, aimed to improve the overall performance and reduce the needed function calls to discover the global minimum. The first modification concerns the application of a sampling technique based on the K-Means method. This technique allowed us to significantly minimize the number of function calls to find the global minimum and further improved the accuracy with which it is located. In particular, the application of the K-Means method accelerated the finding of a solution, as it more efficiently located the points of interest in the search space and led to the fastest convergence to the global minimum. Compared to other methods based on random distributions, the proposed technique proved its superiority, especially in multidimensional and complex functions.

The second proposed amendment concerns the termination rule based on similarity of solutions during iterations. The main purpose of this rule is to stop the optimization process when the iterated solutions are too close to each other, thus preventing pointless iterations that do not provide any significant improvement. It therefore avoids wasting computing time in cases where the process is already very close to the desired result. The use of the termination rule significantly improves the efficiency of the algorithm.

Furthermore, one more improvement was suggested in this research paper. This optimization adds randomness to the generation of new candidate solutions from the old ones aiming to better explore the search space of the objective problem in search of the global minimum.

To verify the effectiveness of the new method, a series of experiments were performed on a large group of objective problems from the recent literature. In these experiments both the efficiency and speed improvement of the original technique was measured and a comparison of the speed of the new method was made in relation to other known techniques from the relevant literature. Furthermore, a series of extensive experiments were carried out to study the dynamics of the proposed initialization technique as well as the additional time required by its implementation. A number of useful conclusions were drawn from

the execution of these experiments. First of all, the new method significantly improves the efficiency and speed of the original method. Furthermore, the experiments revealed that the new method requires a significantly lower number of function calls on average than other global optimization methods. In addition, the sampling method proved to be highly efficient in finding the global minimum and significantly reduced the required number of function calls compared to other initialization techniques. The additional time required by the new initialization method is noticeable compared to other techniques but the gains it brings are equally significant.

Future extensions of the proposed technique could be the use of parallel programming techniques to speed up the overall process, such as the MPI programming technique [114] or the integration of the OpenMP library [115], as well as the use of other termination techniques that could potentially speed up the termination of the method.

Author Contributions: G.K., V.C. and I.G.T. created the software. G.K. conducted the experiments, using a series of objective functions from various sources. V.C. conducted the needed statistical tests. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Acknowledgments: This research has been financed by the European Union: Next Generation EU through the Program Greece 2.0 National Recovery and Resilience Plan, under the call RESEARCH–CREATE–INNOVATE, project name “iCREW: Intelligent small craft simulator for advanced crew training using Virtual Reality techniques” (project code: TAEDK-06195).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Törn, A., Ali, M. M., & Viitanen, S. (1999). Stochastic global optimization: Problem classes and solution techniques. Journal of Global Optimization, 14, 437-447.
2. Floudas, C. A., & Pardalos, P. M. (Eds.). (2013). State of the art in global optimization: computational methods and applications
3. Horst, R., & Pardalos, P. M. (Eds.). (2013). Handbook of global optimization (Vol. 2). Springer Science & Business Media.
4. Intriligator, M. D. (2002). Mathematical optimization and economic theory. Society for Industrial and Applied Mathematics.
5. Cánovas, M. J., Kruger, A., Phu, H. X., & Théra, M. (2020). Marco A. López, a Pioneer of Continuous Optimization in Spain. Vietnam Journal of Mathematics, 48, 211-219.
6. Mahmoodabadi, M. J., & Nemati, A. R. (2016). A novel adaptive genetic algorithm for global optimization of mathematical test functions and real-world problems. Engineering Science and Technology, an International Journal, 19(4), 2002-2021.
7. Li, J., Xiao, X., Boukouvala, F., Floudas, C. A., Zhao, B., Du, G., ... & Liu, H. (2016). Data-driven mathematical modeling and global optimization framework for entire petrochemical planning operations. AIChE Journal, 62(9), 3020-3040.
8. Iuliano, E. (2017). Global optimization of benchmark aerodynamic cases using physics-based surrogate models. Aerospace Science and Technology, 67, 273-286.
9. Duan, Q., Sorooshian, S., & Gupta, V. (1992). Effective and efficient global optimization for conceptual rainfall-runoff models. Water resources research, 28(4), 1015-1031.
10. Yang, L., Robin, D., Sannibale, F., Steier, C., & Wan, W. (2009). Global optimization of an accelerator lattice using multiobjective genetic algorithms. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 609(1), 50-57.
11. Heiles, S., & Johnston, R. L. (2013). Global optimization of clusters using electronic structure methods. International Journal of Quantum Chemistry, 113(18), 2091-2109.
12. Shin, W. H., Kim, J. K., Kim, D. S., & Seok, C. (2013). GalaxyDock2: Protein–ligand docking using beta-complex and global optimization. Journal of computational chemistry, 34(30), 2647-2656.
13. Liwo, A., Lee, J., Ripoll, D. R., Pillardy, J., & Scheraga, H. A. (1999). Protein structure prediction by global optimization of a potential energy function. Proceedings of the National Academy of Sciences, 96(10), 5482-5485.
14. Lee, E. K. (2007). Large-scale optimization-based classification models in medicine and biology. Annals of biomedical engineering, 35, 1095-1109.
15. Cherruault, Y. (1994). Global optimization in biology and medicine. Mathematical and computer modelling, 20(6), 119-132.

16. Houssein, E. H., Hosney, M. E., Mohamed, W. M., Ali, A. A., & Younis, E. M. (2023). Fuzzy-based hunger games search algorithm for global optimization and feature selection using medical data. Neural Computing and Applications, 35(7), 5251-5275.
17. Ion, I. G., Bontinck, Z., Loukrezis, D., Römer, U., Lass, O., Ulbrich, S., ... & De Gersem, H. (2018). Robust shape optimization of electric devices based on deterministic optimization methods and finite-element analysis with affine parametrization and design elements. Electrical Engineering, 100(4), 2635-2647.
18. Cuevas-Velásquez, V., Sordo-Ward, A., García-Palacios, J. H., Bianucci, P., & Garrote, L. (2020). Probabilistic model for real-time flood operation of a dam based on a deterministic optimization model. Water, 12(11), 3206.
19. Pereyra, M., Schniter, P., Chouzenoux, E., Pesquet, J. C., Tourneret, J. Y., Hero, A. O., & McLaughlin, S. (2015). A survey of stochastic simulation and optimization methods in signal processing. IEEE Journal of Selected Topics in Signal Processing, 10(2), 224-241.
20. Hannah, L. A. (2015). Stochastic optimization. International Encyclopedia of the Social & Behavioral Sciences, 2, 473-481.
21. Kizielewicz, B., & Sałabun, W. (2020). A new approach to identifying a multi-criteria decision model based on stochastic optimization techniques. Symmetry, 12(9), 1551.
22. Chen, T., Sun, Y., & Yin, W. (2021). Solving stochastic compositional optimization is nearly as easy as solving stochastic optimization. IEEE Transactions on Signal Processing, 69, 4937-4948.
23. **Wolfe, M. A. (1996). Interval methods for global optimization. Applied Mathematics and Computation, 75(2-3), 179-206.**
24. Csendes, T., & Ratz, D. (1997). Subdivision direction selection in interval methods for global optimization. SIAM Journal on Numerical Analysis, 34(3), 922-938.
25. Sergeyev, Y. D., Kvasov, D. E., & Mukhametzhanov, M. S. (2018). On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. Scientific reports, 8(1), 453.
26. Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11, 341-359.
27. Liu, J., & Lampinen, J. (2005). A fuzzy adaptive differential evolution algorithm. Soft Computing, 9, 448-462.
28. Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks (Vol. 4, pp. 1942-1948). ieee.
29. Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization: An overview. Swarm intelligence, 1, 33-57.
30. Trelea, I. C. (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection. Information processing letters, 85(6), 317-325.
31. Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE computational intelligence magazine, 1(4), 28-39.
32. Socha, K., & Dorigo, M. (2008). Ant colony optimization for continuous domains. European journal of operational research, 185(3), 1155-1173.
33. Goldberg, D. E. (1989). Cenetic algorithms in search. Optimization, Machine Learning.
34. Michalewicz, Z. (1999). Genetic Algorithms+ Data Structures= Evolution Programs. Springer-Verlag, 1999. Google Scholar Google Scholar Digital Library Digital Library.
35. Abdel-Basset, M., El-Shahat, D., Jameel, M., & Abouhawwash, M. (2023). Exponential distribution optimizer (EDO): a novel math-inspired algorithm for global optimization and engineering problems. Artificial Intelligence Review, 56(9), 9329-9400.
36. Ma, L., Cheng, S., & Shi, Y. (2020). Enhancing learning efficiency of brain storm optimization via orthogonal learning design. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 51(11), 6723-6742.
37. Zhou, Y., & Tan, Y. (2009, May). GPU-based parallel particle swarm optimization. In 2009 IEEE Congress on Evolutionary Computation (pp. 1493-1500). IEEE.
38. Dawson, L., & Stewart, I. (2013, June). Improving Ant Colony Optimization performance on the GPU using CUDA. In 2013 IEEE Congress on Evolutionary Computation (pp. 1901-1908). IEEE.
39. Barkalov, K., & Gergel, V. (2016). Parallel global optimization on GPU. Journal of Global Optimization, 66, 3-20.
40. Hassanien, A. E., & Emary, E. (2018). Swarm intelligence: principles, advances, and applications. CRC press.
41. Tang, J., Liu, G., & Pan, Q. (2021). A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends. IEEE/CAA Journal of Automatica Sinica, 8(10), 1627-1643.
42. Brezočnik, L., Fister Jr, I., & Podgorelec, V. (2018). Swarm intelligence algorithms for feature selection: a review. Applied Sciences, 8(9), 1521.
43. Tang, J., Liu, G., & Pan, Q. (2021). A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends. IEEE/CAA Journal of Automatica Sinica, 8(10), 1627-1643.
44. Chu, Y., Mi, H., Liao, H., Ji, Z., & Wu, Q. H. (2008, June). A fast bacterial swarming algorithm for high-dimensional function optimization. In 2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence) (pp. 3135-3140). IEEE.
45. Neshat, M., Sepidnam, G., Sargolzaei, M., & Toosi, A. N. (2014). Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. Artificial intelligence review, 42(4), 965-997.
46. Wu, T. Q., Yao, M., & Yang, J. H. (2016). Dolphin swarm algorithm. Frontiers of Information Technology & Electronic Engineering, 17(8), 717-729.
47. Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. Advances in engineering software, 95, 51-67.

48. Nasiri, J., & Khiyabani, F. M. (2018). A whale optimization algorithm (WOA) approach for clustering. Cogent Mathematics & Statistics, 5(1), 1483565.
49. Gharehchopogh, F. S., & Gholizadeh, H. (2019). A comprehensive survey: Whale Optimization Algorithm and its applications. Swarm and Evolutionary Computation, 48, 1-24.
50. Wang, J., Bei, J., Song, H., Zhang, H., & Zhang, P. (2023). A whale optimization algorithm with combined mutation and removing similarity for global optimization and multilevel thresholding image segmentation. Applied Soft Computing, 137, 110130.
51. Kaur, S., Awasthi, L. K., Sangal, A. L., & Dhiman, G. (2020). Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. Engineering Applications of Artificial Intelligence, 90, 103541.
52. Mirjalili, S. (2016). SCA: a sine cosine algorithm for solving optimization problems. Knowledge-based systems, 96, 120-133.
53. Hao, Y., Song, L., Cui, L., & Wang, H. (2019). A three-dimensional geometric features-based SCA algorithm for compound faults diagnosis. Measurement, 134, 480-491.
54. Sahu, P. C., Prusty, R. C., & Panda, S. (2022). Optimal design of a robust FO-Multistage controller for the frequency awareness of an islanded AC microgrid under i-SCA algorithm. International Journal of Ambient Energy, 43(1), 2681-2693.
55. Zivkovic, M., Stoean, C., Chhabra, A., Budimirovic, N., Petrovic, A., & Bacanin, N. (2022). Novel improved salp swarm algorithm: An application for feature selection. Sensors, 22(5), 1711.
56. Wan, Y., Mao, M., Zhou, L., Zhang, Q., Xi, X., & Zheng, C. (2019). A novel nature-inspired maximum power point tracking (MPPT) controller based on SSA-GWO algorithm for partially shaded photovoltaic systems. Electronics, 8(6), 680.
57. Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in engineering software, 114, 163-191.
58. Bairathi, D., & Gopalani, D. (2019). Salp swarm algorithm (SSA) for training feed-forward neural networks. In Soft Computing for Problem Solving: SocProS 2017, Volume 1 (pp. 521-534). Springer Singapore.
59. Abualigah, L., Shehab, M., Alshinwan, M., & Alabool, H. (2020). Salp swarm algorithm: a comprehensive survey. Neural Computing and Applications, 32(15), 11195-11215.
60. Abdullahi, M., Ngadi, M. A., Dishing, S. I., Abdulhamid, S. I. M., & Usman, M. J. (2020). A survey of symbiotic organisms search algorithms and applications. Neural computing and applications, 32(2), 547-566.
61. Ezugwu, A. E., Adeleke, O. J., Akinyelu, A. A., & Viriri, S. (2020). A conceptual comparison of several metaheuristic algorithms on continuous optimisation problems. Neural Computing and Applications, 32(10), 6207-6251.
62. Wang, Y., & DeAngelis, D. L. (2012). A mutualism-parasitism system modeling host and parasite withmutualism at low density. Mathematical Biosciences & Engineering, 9(2), 431-444.
63. Aubier, T. G., Joron, M., & Sherratt, T. N. (2017). Mimicry among unequally defended prey should be mutualistic when predators sample optimally. The American Naturalist, 189(3), 267-282.
64. Addicott, J. F. (1985). Competition in mutualistic systems. The biology of mutualism: ecology and evolution. Croom Helm, London, UK, 217-247.
65. Bshary, R., Hohner, A., Ait-el-Djoudi, K., & Fricke, H. (2006). Interspecific communicative and coordinated hunting between groupers and giant moray eels in the Red Sea. PLoS biology, 4(12), e431.
66. Ali Mohammadzadeh, Seyedali Mirjalili. (2024) Eel and grouper optimizer: a nature-inspired optimization algorithm. Springer Science+Business Media, LLC, part of Springer Nature 2024
67. Gogu, A., Nace, D., Dilo, A., Meratnia, N., & Ortiz, J. H. (2012). Review of optimization problems in wireless sensor networks. Telecommunications Networks—Current Status and Future Trends, 153-180.
68. Goudos, S. K., Boursianis, A. D., Mohamed, A. W., Wan, S., Sarigiannidis, P., Karagiannidis, G. K., & Suganthan, P. N. (2021, July). Large Scale Global Optimization Algorithms for IoT Networks: A Comparative Study. In 2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS) (pp. 272-279). IEEE.
69. Arayapan, K., & Warunyuwong, P. (2009). Logistics optimization: Application of optimization modeling in inbound logistics.
70. Singh, S. P., Dhiman, G., Juneja, S., Viriyasitavat, W., Singal, G., Kumar, N., & Johri, P. (2023). A New QoS Optimization in IoT-Smart Agriculture Using Rapid Adaption Based Nature-Inspired Approach. IEEE Internet of Things Journal.
71. Wang, H., & Ersoy, O. K. (2005). A novel evolutionary global optimization algorithm and its application in bioinformatics. ECE Technical Reports, 65.
72. Cassioli, A., Di Lorenzo, D., Locatelli, M., Schoen, F., & Sciandrone, M. (2012). Machine learning for global optimization. Computational Optimization and Applications, 51, 279-303.
73. Houssein, E. H., Helmy, B. E. D., Elngar, A. A., Abdelminaam, D. S., & Shaban, H. (2021). An improved tunicate swarm algorithm for global optimization and image segmentation. IEEE Access, 9, 56066-56092.
74. Torun, H. M., & Swaminathan, M. (2019). High-dimensional global optimization method for high-frequency electronic design. IEEE Transactions on Microwave Theory and Techniques, 67(6), 2128-2142.
75. Wang, L., Kan, J., Guo, J., & Wang, C. (2019). 3D path planning for the ground robot with improved ant colony optimization. Sensors, 19(4), 815.
76. Arora, P., & Varshney, S. (2016). Analysis of k-means and k-medoids algorithm for big data. Procedia Computer Science, 78, 507-512.
77. Ahmed, M., Seraj, R., & Islam, S. M. S. (2020). The k-means algorithm: A comprehensive survey and performance evaluation. Electronics, 9(8), 1295.

78. Charilogis, V., & Tsoulos, I. G. (2022). Toward an ideal particle swarm optimizer for multidimensional functions. Information, 13(5), 217.
79. Macqueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press.
80. H. Maaranen, K. Miettinen, M.M. Makela, Quasi-random initial population for genetic algorithms. Computers & Mathematics with Applications **47**, pp. 1885-1895, 2004.
81. P.V. Paul, P. Dhavachelvan, R. Baskaran, A novel population initialization technique for Genetic Algorithm, In: 2013 International Conference on Circuits, Power and Computing Technologies (ICCPCT), Nagercoil, India, pp. 1235-1238, 2013.
82. Ali, M., Pant, M., & Abraham, A. (2013). Unconventional initialization methods for differential evolution. Applied Mathematics and Computation, 219(9), 4474-4494.
83. Bajer, D., Martinović, G., & Brest, J. (2016). A population initialization method for evolutionary algorithms based on clustering and Cauchy deviates. Expert Systems with Applications, 60, 294-310.
84. Kazimipour, B., Li, X., & Qin, A. K. (2014, July). A review of population initialization techniques for evolutionary algorithms. In 2014 IEEE congress on evolutionary computation (CEC) (pp. 2585-2592). IEEE.
85. Li, Y., & Wu, H. (2012). A clustering method based on K-means algorithm. Physics Procedia, 25, 1104-1109.
86. B.J. Jain, H. Pohlheim, J. Wegener, On termination criteria of evolutionary algorithms, in: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, 2001, pp. 768–768.
87. K. Zielinski, P. Weitkemper, R. Laur, K.-D. Kammeyer, Examination of stopping criteria for differential evolution based on a power allocation problem, J. Electr. Eng. 7 (3) (2007) 8.
88. S.N. Ghoreishi, A. Clausen, B.N. Joergensen, Termination criteria in evolutionary algorithms: A survey, in: IJCCI, 2017, pp. 373–384.
89. M. Ravber, S-H. Liu, M. Mernik, M. Črepinšek, Maximum number of generations as a stopping criterion considered harmful, Applied Soft Computing, Volume 128, 2022, 109478, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2022.109478.
90. Ali, H. H., & Kadhum, L. E. (2017). K-means clustering algorithm applications in data mining and pattern recognition. International Journal of Science and Research (IJSR), 6(8), 1577-1584.
91. Krishna, K., & Murty, M. N. (1999). Genetic K-means algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 29(3), 433-439.
92. Sinaga, K. P., & Yang, M. S. (2020). Unsupervised K-means clustering algorithm. IEEE access, 8, 80716-80727.
93. Ay, M., Özbakır, L., Kulluk, S., Gülmez, B., Öztürk, G., & Özer, S. (2023). FC-Kmeans: Fixed-centered K-means algorithm. Expert Systems with Applications, 211, 118656.
94. Oti, E. U., Olusola, M. O., Eze, F. C., & Enogwe, S. U. (2021). Comprehensive review of K-Means clustering algorithms. criterion, 12, 22-23.
95. Ali, M. M., Khompatraporn, C., & Zabinsky, Z. B. (2005). A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. Journal of global optimization, 31, 635-672.
96. Floudas, C. A., Pardalos, P. M., Adjiman, C., Esposito, W. R., Gümüs, Z. H., Harding, S. T., ... & Schweiger, C. A. (2013). Handbook of test problems in local and global optimization (Vol. 33). Springer Science & Business Media.
97. M.M. Ali and P. Kael. (2008). Improved particle swarm algorithms for global optimization, Applied Mathematics and Computation **196**, pp. 578-593.
98. H. Koyuncu, R. Ceylan.(2009). A PSO based approach: Scout particle swarm algorithm for continuous global optimization problems, Journal of Computational Design and Engineering **6**, pp. 129–142.
99. Siarry, P., Berthiau, G., Durdin, F., & Haussy, J. (1997). Enhanced simulated annealing for globally minimizing functions of many-continuous variables. ACM Transactions on Mathematical Software (TOMS), 23(2), 209-228.
100. Tsoulos, I. G., & Lagaris, I. E. (2008). GenMin: An enhanced genetic algorithm for global optimization. Computer Physics Communications, 178(11), 843-851.
101. LaTorre, A., Molina, D., Osaba, E., Poyatos, J., Del Ser, J., & Herrera, F. (2021). A prescription of methodological guidelines for comparing bio-inspired optimization algorithms. Swarm and Evolutionary Computation, 67, 100973.
102. Li, X., Engelbrecht, A., & Epitropakis, M. G. (2013). Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization. RMIT University, Evolutionary Computation and Machine Learning Group, Australia, Tech. Rep.
103. Gaviano, M., Kvasov, D. E., Lera, D., & Sergeyev, Y. D. (2003). Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. ACM Transactions on Mathematical Software (TOMS), 29(4), 469-480
104. Jones, J. E. (1924). On the determination of molecular fields.—II. From the equation of state of a gas. Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character, 106(738), 463-477.
105. Powell, M.J.D. (1989). A Tolerant Algorithm for Linearly Constrained Optimization Calculations. Math. Program, 45, 547–566.
106. Tsoulos, I. G. (2008). Modifications of real code genetic algorithm for global optimization. Applied Mathematics and Computation, 203(2), 598-607.
107. Charilogis, V., & Tsoulos, I. G. (2022). Toward an ideal particle swarm optimizer for multidimensional functions. Information, 13(5), 217.

108. Stein, W. E., & Keblis, M. F. (2009). A new method to simulate the triangular distribution. Mathematical and Computer Modelling, 49(5-6), 1143-1147.
109. Sharma, V. K., Bakouch, H. S., & Suthar, K. (2017). An extended Maxwell distribution: Properties and applications. Communications in Statistics-Simulation and Computation, 46(9), 6982-7007.
110. Browne, D. et al., (2022). Applications of Uniform Distributions in Optimization Algorithms. Journal of Applied Statistics, 45(2), 123-139.
111. Zhao, Y., et al., (2023). Risk Management Models Using Triangular Distributions. Risk Analysis Quarterly, 39(1), 98-112.
112. Chen, L., et al., (2021). Simulating Communication Networks with Maxwell Distribution. IEEE Transactions on Communications, 68(4), 568-578.
113. Xu, J., et al., (2022). Improving K-means Clustering with K-means++ Initialization. Machine Learning Journal, 34(3), 245-267.
114. Gropp, W., Lusk, E., Doss, N., & Skjellum, A. (1996). A high-performance, portable implementation of the MPI message passing interface standard. Parallel computing, 22(6), 789-828.
115. Chandra, R. (2001). Parallel Programming in OpenMP. Academic Press.
116. Gad, A. G. (2022). Particle swarm optimization algorithm and its applications: a systematic review. Archives of computational methods in engineering, 29(5), 2531-2561.