Article

An innovative hybrid approach producing trial solutions for Global Optimization

Vasileios Charilogis[1],Glykeria Kyrou[2], Ioannis G. Tsoulos[3,*], Anna Maria Gianni[4]

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

Abstract: Global optimization is critical in engineering, computer science, and various industrial applications, as it aims to find optimal solutions for complex problems. The development of efficient algorithms has emerged from the need for optimization, with each algorithm offering specific advantages and disadvantages. An effective approach to solving complex problems is the hybrid method, which combines established global optimization algorithms. This paper presents a hybrid global optimization method, which produces trial solutions for an objective problem, utilizing Genetic Algorithm genetic operators, as well as solutions obtained through a linear search process. Then, the generated solutions are used to form new test solutions, by applying Differential Evolution techniques. These operations are based on samples derived either from internal line searches or genetically modified samples in specific subsets of Euclidean space. Additionally, other relevant approaches are explored to enhance the method's efficiency. The new method was applied on a wide series of benchmark problems from the recent literature and comparison was made against other established methods of Global Optimization.

Keywords: Optimization; Differential evolution; Genetic algorithm; Line search; Evolutionary techniques; Stochastic methods; Hybrid methods.

Citation: Charilogis V.; Kyrou, G.;Tsoulos I.G.; Gianni A.M; An Innovative Hybrid Approach to Global Optimization. *Journal Not Specified* **2023**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

1. Introduction

The primary objective of global optimization is to locate the global minimum by thoroughly exploring the relevant range associated with the underlying objective problem. This method of global optimization is focused on identifying the global minimum within a continuous function that spans multiple dimensions. Essentially, the global optimization process is dedicated to seeking out the minimum value of a continuous, multidimensional function, ensuring that the search covers all potential ranges of the problem at hand. The objective is to find the lowest point through systematic exploration of the entire domain of the function$f : S \to R, S \subset R^n$ and it is defined as follows:

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

where the set S is defined as follows:

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

Global optimization refers to algorithms that aim to find the overall optimum for an objective function. According to the literature survey, there are a variety of real-world problems that can be applied in mathematics. For example in the optimization of classification and regression trees[1], in physics for optimization of quantum devices with noise[2], in chemistry for optimization of chemical structures clustering[3], in medicine for feature selection using medical data [4], in biology for the application of artificial intelligence models and optimization algorithms in plant cell and tissue culture[5], in agriculture in sugarcane production applications[6] and economics for comprehensive techno-economic analysis[7]. Optimization methods can be categorized into deterministic[8] and stochastic[9] based on how they approach solving the problem. The techniques belonging to the category of deterministic methods are mainly the so - called interval methods[10]. Stochastic methods utilize randomness to explore the solution space, while in interval methods, the set S is divided into smaller regions that may contain the global minimum based on certain criteria. Recently, a comparison between deterministic and stochastic methods was proposed by Sergeyev et al. [11].

A series of stochastic optimization methods are the so - called evolutionary methods, which attempt to mimic a series of natural processes. Such methods include the Genetic algorithms[12], the Differential Evolution method [13], the Particle Swarm Optimization (PSO) method used, for example, in solving feature selection problems[14], the Ant Colony Optimization method used for example in the traveling salesman problem[15], the Fish Swarm Algorithm which is inspired by the ecological behaviors of fish breeding in the wild, i.e. prey, swarming and monitoring behaviors.[16], the Dolphin Swarm Algorithm based on the natural behavior of sparrows and dolphins. [17], the Whale Optimization Algorithm (WOA) algorithm[18] etc. Also, due to the wide spread of parallel computing units[19], a variety of research papers related to evolutionary techniques appeared that use such processing units[20].

Genetic algorithms were formulated by John Holland[21] and his team, and they form randomly candidate solutions for the optimization problem. These solutions are modified through a series of operators that mimic natural processes, such as mutation, selection and crossover. Genetic algorithms have been widely used in areas such as networking for indoor 5G network design using a machine learning model for path loss estimation[22], robotics for digital twin robots[23], energy conservation issues of a high SI motor speed fueled with butanol-gasoline blends[24] etc. They can be combined with machine learning to solve complex problems, such as training a feed-forward artificial neural network[25].

Furthermore, differential evolution (DE) is used in symmetric optimization problems[26] and in problems that are discontinuous and noisy and change over time. After studies, it was observed that differential evolution can be successfully combined with other techniques for machine learning applications, such as image classification[27], feature selection, predicting heart disease problems[28], compression of deep neural networks [29] etc.

Hybrid methods [30] in global optimization refer to techniques that combine multiple optimization strategies to solve complex problems. These methods aim to take advantage of different approaches to find the global optimum in a more efficient way, particularly when dealing with large-scale problems or strongly nonlinear optimization landscapes. A typical example of a hybrid method is the work of Shutao Li et al. who proposed a new hybrid PSO-BFGS strategy for the global optimization of multimodal functions [31]. To make the combination more efficient, they proposed an LDI to dynamically start the local search and a repositioning technique to maintain the particle diversity, which can effectively avoid the premature convergence problem. Another innovative hybrid method is the work of M. Andalib Sahnehsaraei et al. where a hybrid algorithm using GA operators and PSO formula is proposed was presented through the use of efficient operators, for example, traditional and multiple crossovers, mutation and PSO formula [32].

In the current work, two evolutionary methods were incorporated into the final algorithm: Genetic Algorithms and the Differential Evolution method were combined into a hybrid optimization method. More specifically, through a series of steps, trial solutions are generated using the genetic operators of the Genetic Algorithm as well as solutions determined by a line search procedure. Additionally, an Armijo line search method is used. This method is incorporated to estimate an appropriate step when updating the trial points, and it was introduced in the work of Armijo [33]. The solutions produced in the previous step are used to formulate new trial solutions using a process derived from Differential Evolution.

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

2. The overall algorithm

Modern optimization methods have been developed to solve complex problems. The GA, inspired by natural selection, uses operators like crossover and mutation but often suffers from slow convergence and high computational cost in high-dimensional problems. PSO leverages particle collaboration to search for optimal solutions, although it can get trapped in local optima. The Improved PSO (IPSO) [34]addresses this issue with dynamic adjustments, enhancing both speed and accuracy. DE stands out for its simplicity and efficiency through mutation and recombination, but its performance relies heavily on proper parameter tuning. The Grey Wolf Optimizer (GWO) [35] mimics the hunting behavior of wolves, balancing exploration and exploitation, but struggles with high-dimensional problems. The WOA, inspired by whales' hunting strategies, has proven effective, though its performance decreases in large-scale problems. The Multi-Enhanced WOA (MEWOA) introduces additional strategies to avoid local optima and accelerate convergence. Each method offers advantages and limitations, with the choice depending on the problem's requirements and available resources.

The proposed algorithm integrates features from different optimization methods to enhance efficiency. It begins with the initialization of the population, which is generated either through uniform distribution or the k-means method. Next, samples are computed based on Armijo's linear search, incorporating elements from Genetic Algorithms while simultaneously applying the DE formula. Specifically, the nearest sample c_i is identified from the initial set x_i as shown in step 2 of the used algorithm. Then, a second sample is generated using the Armijo linear search, following the termination criterion of the method, considering both the initial and the nearest sample. A third sample is subsequently computed through the "crossover" mechanism of Genetic Algorithms , combining the initial sample with the best one obtained from the entire search space. These samples are used to create a new trial sample based on the formula 5, leveraging both local information and the position of the best sample. The periodic application of a local optimization method further improves the algorithm's performance by ensuring solution refinement. The termination criterion is controlled through an adaptive process, determining whether the search should continue or stop based on the achieved progress. In the conducted tests, the algorithm demonstrated high efficiency by significantly reducing the number of objective function evaluations and the overall computational cost. The algorithm is as follows:

1. **Initialization step.**
   - (a) **Set** the population size $N \geq 4$.
   - (b) **Set** n the dimension of the benchmark function.
   - (c) **Initialize** the samples $x_i, i = 1, \ldots, N$ using uniform or k-means [36] distribution.
2. **Calculation step.**
   - (a) **For** $i = 1 \ldots \mathrm{N}$ **do**
     - i. **Obtain** sample x_i.
     - ii. **Find** nearest sample c_i from x_i**:**

$$d(x_i, c_i) = \sqrt{\sum_{j=1}^{n} (x_{i,j} - c_{i,j})^2} \tag{2}$$

     - iii. **Set** direction vectors: $p_1 = -\nabla fs(x_i)$ end $p_2 = -\nabla fs(c_i)$
     - iv. **Set** initial step size for Armijo $a = a_0$

v. **Find** new points using line search Armijo minLS(x_i, c_i): $x_i^{new} = x_i + ap_1$ and $c_i^{new} = c_i + ap_2$

vi. **Set** the reduction constant $c_1 = 10^{-4}$

vii. **Adjust** step size a until Armijo condition is met:

$$fs(x_i^{new}, c_i{}^{new}) \leq fs(x_i, c_i) + c_1 a \nabla fs(x_i, c_i)^T (p_1 + p_2) \tag{3}$$

viii. **Make** sample-child with crossover with random number $g_k \in [0, 1]$:

$$\text{child}\left(x_i, x^{best}\right) = g_k x_{i,k} + (1 - g_k) x_k^{best} \tag{4}$$

ix. **For** $j = 1, \ldots, n$ **do**

- **Set** trial vector:

$$y_j = x_{i,j} + F \times \left(\text{minLS}(x_\iota, c_\iota)_j - \text{child}(x_i, c_i)_j\right) \tag{5}$$

  where F is the so - called Differential Weight of Differential Evolution algorithm.

- **If** $y_j \notin \left[a_j, b_j\right]$, then $y_j = x_{i,j}$

x. **EndFor**

- **Set** $r \in [0, 1]$ a random number. If $r \leq p_m$ then $x_i = \text{LS}(x_i)$, where $\text{LS}(x)$ is a local search procedure, such as the BFGS procedure [37].
- **If** $f(y) \leq f(x)$ then $x = y$, $x^{best} = y$.

(b) **EndFor**

3. **Check the termination rule stated in** [34], which means that the method checks the difference between the current optimal solution $f_{min}^{(t)}$ and the previous $f_{min}^{(t-1)}$ one. The algorithm terminates when the following:

$$\left|f_{\text{min}}^{(t)} - f_{\text{min}}^{(t-1)}\right| \leq \epsilon \tag{6}$$

   holds for N_t iterations. The value ϵ is a small positive value. In the conducted experiments the value $\epsilon = 10^{-5}$ was used. If the termination rule of equation 6 does not hold, then the algorithm continues from Step 2.

4. **Return** the sample x^{best} in the population with the lower function value $f\left(x^{best}\right)$.

The algorithm is also shown as a flowchart in Figure 1.

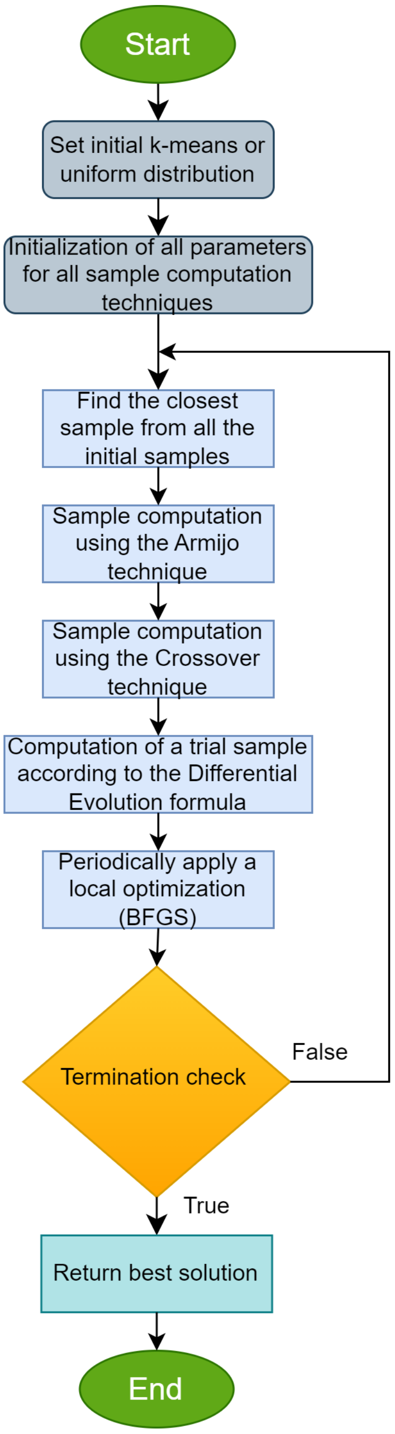

Start
Set initial k-means or uniform distribution
Initialization of all parameters for all sample computation techniques
Find the closest sample from all the initial samples
Sample computation using the Armijo technique
Sample computation using the Crossover technique
Computation of a trial sample according to the Differential Evolution formula
Periodically apply a local optimization (BFGS)
Termination check
False
True
Return best solution
End

Figure 1. The flowchart of the proposed optimization procedure.

The optimization method described here combines evolutionary techniques, such as differential evolution, Armijo line search, and components of genetic algorithms, with the aim of finding the optimal solution. Initially, the population size and the dimensionality of the target function are defined, and the population samples are generated randomly using a uniform or k-means distribution. For each sample, the Euclidean distance (equation 2) to the other samples is calculated to identify the nearest one, followed by an Armijo line search (equation 3) to determine the optimal movement direction for the initial sample. Subsequently, a new offspring sample is created using a crossover process (equation 4), combining the current sample with the best discovered so far. A trial vector (equation 5) is then formed, which accounts for the adjustment of the computed samples and the differential coefficient parameter derived from the differential evolution algorithm. Periodically, local search methods, such as BFGS [37], are applied to improve the accuracy of the solution search. The method terminates when the best solution found remains nearly unchanged for a specified number of iterations. In summary, the basic steps for calculating a new sample are:

- Identification of the nearest point c_ι for each sample x_ι.
- Calculation of a sample minLS(x_i, c_i) through Armijo line search, between the sample x_i and the samplec_i.
- Generation of the sample using the crossover process of the Genetic Algorithm, between the sample x_ι and the best samplex^{best} .
- Computation of the trial point y_i using a process derived from Differential Evolution.

3. Experiments

Settings and benchmark functions

The benchmark functions used in the experimental measurements are presented in Table 1.

Table 1. The benchmark functions used in the conducted experiments.

NAME	FORMULA	DIMENSION
ACKLEY	$f(x) = -a\exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(cx_i)\right) + a + \exp(1) \quad a = 20.0$	2
BF1	$f(x) = -20\exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(cx_i)\right) + 20 + \exp(1)$	2
BF1	$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$	2
BF2	$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$	2
BF3	$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$	2
DIFFPOWER	$f(x) = \sum_{i=1}^{n} \lvert x_i - y_i\rvert^p$	$n = 2\ p = 2, 5, 10$
CAMEL	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, \quad x \in [-5, 5]^2$	2
EASOM	$f(x) = -\cos(x_1)\cos(x_2)\exp\left((x_2 - \pi)^2 - (x_1 - \pi)^2\right)$	2
ELP	$f(x) = \sum_{i=1}^{n} (10^6)^{\frac{i-1}{n-1}} x_i^2$	$n = 10, 20, 30$
EXP	$f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right), \quad -1 \le x_i \le 1$	$n = 4, 8, 16, 32$
F3	$f(x) = \left(e^{-2.0\log(2.0)\left(\frac{(x_1-0.08)}{0.854}\right)^2}\right)\left(\sin\left(5.0\pi\left(x_1^{\frac{3.0}{4.0}} - 0.05\right)\right)\right)^6 \quad x \in [0,1]^n$	2
F5	$f(x) = \left(\left(4.0 - 2.1x_1^2 + \frac{x_1^4}{3.0}\right)x_1^2\right) + (x_1x_2) + \left((4.0x_2^2 - 4.0)x_2^2\right) \quad -5 \le x_i \le 5$	2
F9	$f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right), \quad x \in [0,1]^n$	2
GKLS[38]	$f(x) = \mathrm{Gkls}(x, n, w)$	$n = 2, 3\ w = 50, 100$
GRIEWANK2	$f(x) = 1 + \frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2}\frac{\cos(x_i)}{\sqrt{(i)}}$	2
GRIEWANK10	$\mathrm{f}(x) = 1 + \frac{1}{200}\sum_{i=1}^{10} x_i^2 - \prod_{i=1}^{10}\frac{\cos(x_i)}{\sqrt{(i)}}$	10
HANSEN	$f(x) = \sum_{i=1}^{5} i\cos[(i-1)x_1 + i]\sum_{j=1}^{5} j\cos[(j+1)x_2 + j]$	2
HARTMAN3	$f(x) = -\sum_{i=1}^{4} c_i\exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	3
HARTAMN6	$f(x) = -\sum_{i=1}^{4} c_i\exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6
POTENTIAL[39]	$V_{LJ}(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^6\right]$	$n = 9, 15, 21, 30$
RARSTIGIN	$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)$	2
ROSENBROCK	$f(x) = \sum_{i=1}^{n-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right), \quad -30 \le x_i \le 30$	$n = 4, 8, 16$
SCHWEFELH	$f(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	2
SCHWEFELH221	$f(x) = 418.9829n + \sum_{i=1}^{n} -x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	2
SCHWEFELH222	$f(x) = \sum_{i=1}^{n-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right), \quad -30 \le x_i \le 30$	2
Shekel5	$f(x) = -\sum_{i=1}^{5}\frac{1}{(x-a_i)(x-a_i)^T + c_i}$	4
Shekel7	$f(x) = -\sum_{i=1}^{7}\frac{1}{(x-a_i)(x-a_i)^T + c_i}$	4
Shekel10	$f(x) = -\sum_{i=1}^{10}\frac{1}{(x-a_i)(x-a_i)^T + c_i}$	4
Sinusoidal[40]	$f(x) = -\left(2.5\prod_{i=1}^{n}\sin(x_i - z) + \prod_{i=1}^{n}\sin(5(x_i - z))\right), \quad 0 \le x_i \le \pi$	$n = 4, 8$
Test2N	$f(x) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i$	$n = 4, 5, 7$
Test30N	$\frac{1}{10}\sin^2(3\pi x_1)\sum_{i=2}^{n-1}\left((x_i - 1)^2\left(1 + \sin^2(3\pi x_{i+1})\right)\right) + (x_n - 1)^2\left(1 + \sin^2(2\pi x_n)\right)$	$n = 3, 4$

The functions used in the conducted experiments have been proposed by various researchers [41,42] in the relevant literature. For a more accurate comparison of the methods, efforts were made to maintain certain parameter values at equal or similar levels. The values for the parameters of the algorithm are presented in Table 2, along with some explanation of each parameter.

Table 2. Parameters of optimization methods settings

PARAMETER	VALUE	EXPLANATION
N	200	Number of samples for all methods
N_k	200	Maximum number of iterations for all methods
SR	$\left\lvert f_{\min}^{(k)} - f_{\min}^{(k-1)} \right\rvert$	Best fitness: Stopping rule for all methods
N_t	12	Similarity max count for all methods
LS	0.05 (5%)	Local search rate for all methods
F	0.8	Differential weight for Differential Evolution
CR	0.9	Crossover Probability for Differential Evolution
C_1, C_2	0.5	Parameters of PSO
G_c	0.1 (10%)	Crossover rate for Genetic Algorithm
G_m	0.05 (5%)	Mutation rate for Genetic Algorithm
M_k	10	Number of samples of sub-population for MEWOA
M_r	1	Number of iterations of exchange information for MEWOA

Experimental results

In the random number generator, different seeds were used to ensure the reliability of the experimental results, with the experiments being repeated 30 times. This process of repetition aims to minimize the likelihood of random errors and to enhance the validity of the results. The experiments were conducted on a system with an AMD Ryzen 5950X processor and 128 GB of RAM, operating in a Linux-Debian environment. Additionally, the open-source optimizer "GlobalOptimus" was used, which is a fully developed optimizer and is available for distribution via the link: http://www.github.com/itsoulos/GlobalOptimus (accessed on 7 November 2024). This optimization software can be installed in any operating system (Linux, Windows, FreeBsd) when the Qt library is available. The software includes a variety of optimization methods as swll as some scripts to automate the optimization process under Windows or Linux. Also, the software provides a graphical user interface under the name Xoptimus and an installation file without dependencies for Windows system is also included. In Table 3, the average function calls for each method and every objective function is presented. The following notation is used in the experimental tables:

1. The column FUNCTION represents the name of the used objective problem.
2. The column GENETIC represents the application of the Genetic Algorithm to the objective problem [12].
3. The PSO column represents the implementation of the classical PSO algorithm as suggested by the literature [14].
4. The column IPSO stands for the application of the Improved PSO algorithm as suggested by Charilogis and Tsoulos [34] to the objective problems.
5. The column DE represents the average function calls for the Differential Evolution optimization technique [43].
6. The column GWO represents the average function calls for the Gray Wolf optimization technique [35].
7. The column WOA represents the average function calls for the Whale Optimization technique [18].

8. The column MEWOA stands for the application of the Improved WOA algorithm as suggested by Shen Y. et al [44] to the objective problems.
9. The total sum of calls for each method is listed at the end of the table.
10. The column PROPOSED stands for the application of the current method.
11. The column KMEANS denotes the incorporation of KMEANS sampling in the current method.

Table 3. Comparison of average function calls of proposed method against others

FUNCTION	GEN	PSO	IPSO	DE	GWO	WOA	MEWOA	PROPOSED	KMEANS
ACKLEY	6749	6885	3418	16183	20182	24766	13879	5818	3490
BF1	4007	4113	1814	8268	9120	9924	7987	5585	3023
BF2	3793	3747	1759	7913	8731	9597	6262	5008	2693
BF3	3479	3305	1689	6327	7791	20117	5893	4282	2442
BRANIN	2376	2522	1730	4101	6055	5939	3367	3026	2087
CAMEL	2869	2908	1754	5609	6688	5917	4756	3327	2607
DIFFPOWER2	5443	8657	2462	14669	22059	11436	11536	8864	5617
DIFFPOWER5	18552	24894	4446	39018	49769	42095	31372	26065	17221
DIFFPOWER10	18801	32534	4690	46914	54442	56381	38559	29549	26615
EASOM	1958	1998	1753	2978	4615	3153	2455	2755	1402
ELP10	3131	4397	1720	6288	9415	10317	5255	4020	3593
ELP20	6160	6883	1988	10794	14173	17040	7722	6537	6194
ELP30	9576	9438	2100	14172	18245	23532	9776	8270	3593
EXP4	2946	3177	1677	5166	6648	5567	4392	3570	2238
EXP16	3250	3477	1660	6498	6978	9968	4624	3854	3588
EXP32	3561	3728	1647	7606	7419	11925	4508	3852	3691
F3	4074	3420	1979	7323	10816	11717	5821	6444	5434
F5	1630	2055	1376	2814	48377	7393	2117	2697	2199
F9	2494	2707	1879	4535	5502	4479	3534	4406	1968
GKLS250	2280	2495	1623	3834	6127	4978	3665	2315	1657
GKLS350	2612	2658	1570	3919	6919	7226	3692	2265	1335
GOLDSTEIN	3687	3856	1886	6781	9312	9016	5287	3930	2727
GRIEWANK2	4500	3168	2299	7429	9253	7948	7285	3850	2223
GRIEWANK10	6409	7942	2142	18490	14708	51143	12341	7551	6593
HANSEN	3209	2892	2204	4185	7253	10498	4116	4565	3092
HARTMAN3	2751	3103	1801	5190	7293	8440	3982	3145	2100
HARTMAN6	3219	3688	1862	5968	8814	22615	4416	4105	2921
POTENTIAL3	4351	5154	3096	6218	11256	9232	5071	4790	4034
POTENTIAL5	7704	10128	6353	9119	22136	39455	8807	7995	7991
POTENTIAL6	10177	11780	7593	10509	32630	46466	11341	8906	9333
POTENTIAL10	13357	16550	10413	12721	38337	59487	18573	12757	13611
RASTRIGIN	4106	3539	2211	6216	9043	8116	5523	3672	2364
ROSENBROCK4	3679	5858	2025	8452	8078	10759	7750	5642	3641
ROSENBROCK8	5269	7843	2085	11530	11167	14094	9469	7926	6119
ROSENBROCK16	8509	11450	2432	17432	15590	23544	13554	11285	7490
SCHWEFEL	1880	2178	1504	3437	4564	4158	3005	2208	1572
SCHWEFEL221	2666	2529	1875	3909	71847	8519	3815	2867	2256
SCHWEFEL222	33724	42897	28965	86861	101960	114161	65835	78734	18058
SHEKEL5	3325	3886	2018	6662	10043	17010	5359	5322	2657
SHEKEL7	3360	4009	1994	6967	10079	17873	5344	5503	2648
SHEKEL10	2488	3985	1914	6757	10055	15524	5321	4784	2901
SINU4	2990	3409	1961	5953	8132	10150	4874	3446	2254
SINU8	3441	3995	1853	6973	10481	20782	5586	4371	3359
SINU16	4320	4680	2251	6979	15356	34409	7274	6450	5992
TEST2N4	3330	3390	1930	6396	8093	11876	4930	3569	2204
TEST2N5	4000	3604	1982	6271	8812	16256	5045	3877	2575
TEST2N7	4775	4020	2157	7074	10124	25005	6363	4285	2831
TEST30N3	3210	4018	2432	6178	7647	8163	5311	4215	4389
TEST30N4	3678	4504	3354	7006	10656	13596	7069	5013	2901
TOTAL	**261106**	**317168**	**145908**	**506409**	**802608**	**916996**	**419939**	**361454**	**230033**
MAX	**33724**	**42897**	**28965**	**86861**	**101960**	**114161**	**65835**	**78734**	**26615**
MIN	**1630**	**1998**	**1376**	**2814**	**4564**	**3153**	**2117**	**2208**	**1335**
STDEV	**5517.7**	**7735.58**	**4121.63**	**13674.61**	**18893.71**	**19622.90**	**10608.07**	**11562.76**	**5885.97**
AVERAGE	**5466.43**	**6613.33**	**3047.47**	**10665.14**	**16791.63**	**19219.63**	**8853.43**	**7495.35**	**5316.8**
MEDIAN	**3678**	**3886**	**1982**	**6781**	**9415**	**11717**	**5359**	**4406**	**2921**
SKEW	**3.54**	**3.35**	**5.55**	**4.32**	**2.89**	**2.9**	**4.02**	**5.3**	**2.53**
KURT	**14.75**	**12.05**	**34.01**	**21.19**	**9.12**	**10.87**	**18.43**	**31.3**	**5.77**

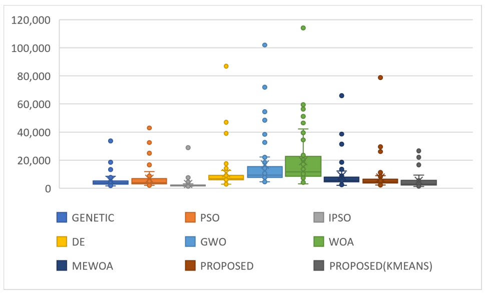
Table 4. Function calls with different differential weight F

FUNCTION	Proposed(F:random)	Proposed (F:0.5)	Proposed (F:0.9)
ACKLEY	5818	5206	5379
BF1	5585	5797	5575
BF2	5008	4666	4734
BF3	4282	4309	4077
BRANIN	3026	3078	2858
CAMEL	3327	3253	3203
DIFFPOWER2	8864	8833	8834
DIFFPOWER5	26065	26067	26059
DIFFPOWER10	29549	29548	29537
EASOM	2755	2668	2637
ELP10	4020	4020	4020
ELP20	6537	6537	6537
ELP30	8270	8270	8270
EXP4	3570	3570	3570
EXP16	3854	3854	3854
EXP32	3852	3852	3852
F1	6444	5649	5617
F5	2697	2551	2328
F9	4406	4063	3500
GKLS250	2315	2321	2316
GKLS350	2265(96)	2244(90)	2204(90)
GOLDSTEIN	3930	3848	3917
GRIEWANK2	3850(50)	3963(50)	4052(50)
GRIEWANK10	7551	7551	7551
HANSEN	4565	4187	3548
HARTMAN3	3145	3142	3139
HARTMAN6	4105	4070	4001
POTENTIAL3	4790	4653	4699
POTENTIAL5	7995	7987	7690
POTENTIAL6	8906(76)	8926(70)	8936(70)
POTENTIAL10	12757	12457	12452
RASTRIGIN	3672(86)	3358(80)	3206(86)
ROSENBROCK4	5642	5641	5641
ROSENBROCK8	7926	7926	7926
ROSENBROCK16	11285	11285	11285
SCHWEFEL	2208	2208	2208
SCHWEFEL221	2867	2858	2742
SCHWEFEL222	78734	89919	90250
SHEKEL5	5322	4721	4492
SHEKEL7	5503	4723	4666
SHEKEL10	4784	4385	4541
SINU4	3446	3634	3488
SINU8	4371	4373	4387
SINU16	6450	6543	6538
TEST2N4	3569	3487	3490
TEST2N5	3877(93)	3704(96)	3712(96)
TEST2N7	4285(50)	4043(50)	4190(53)
TEST30N3	4215	3412	3685
TEST30N4	5013	4812	4515
SUM	361454	366966	342529

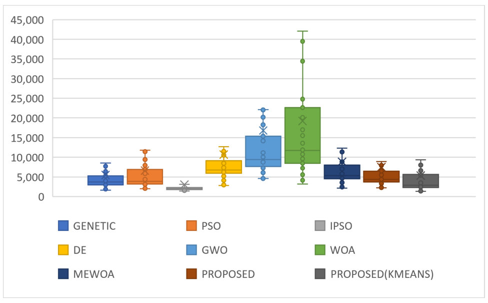
Table 5. Function calls with different number of population

FUNCTION	Proposed (population=200)	Proposed (population=150)	Proposed (population=100)
ACKLAY	5818	4647	2829
BF1	5585	4303	2847
BF2	5008	3502	2334
BF3	4282	3034	2045
BRANIN	3026	2300	1506
CAMEL	3327	2526	1795
DIFFPOWER2	8864	6326	4094
DIFFPOWER5	26065	19239	14008
DIFFPOWER10	29549	22112	15325
EASOM	2755	1976	1306
ELP10	4020	3080	2111
ELP20	6537	5080	3234
ELP30	8270	6300	4307
EXP4	3570	2718	1906
EXP16	3854	2938	1990
EXP32	3852	2700	1904
F1	6444	5232	3404
F5	2697	2100	1822
F9	4406	2999	2111
GKLS250	2315	1767	1191
GKLS350	2265(96)	1699 (90)	1050(60)
GOLDSTEIN	3930	2826	1949
GRIEWANK2	3850(50)	2803(56)	1883(30)
GRIEWANK10	7551	5821	3933
HANSEN	4565	2965	2024
HARTMAN3	3145	2440	1613
HARTMAN6	4105	3036	2004
POTENTIAL3	4790	3655	2416
POTENTIAL5	7995	6710	4215
POTENTIAL6	8906(76)	8246(70)	4890(33)
POTENTIAL10	12757	10417	6598(86)
RASTRIGIN	3672(86)	2525(73)	1728(56)
ROSENBROCK4	5642	4343	2995
ROSENBROCK8	7926	5953	4083
ROSENBROCK16	11285	8694	5922
SCHWEFEL	2208	1728	1125
SCHWEFEL221	2867	2192	1516
SCHWEFEL222	78734	64540	30545
SHEKEL5	5322	3863	2407
SHEKEL7	5503	3568	2556
SHEKEL10	4784	3277	2374
SINU4	3446	2561	1858
SINU8	4371	3292	2258
SINU16	6450	5162	3386
TEST2N4	3569	2597	1988(86)
TEST2N5	3877(93)	3130(96)	1915(80)
TEST2N7	4285(50)	3325(36)	2089(33)
TEST30N3	4215	2756	1870
TEST30N4	5013	3768	2115
SUM	361454	278124	174545

At the end of every global optimization procedure, a BFGS variant of Powell [45] is utilized to improve the discovered solution and identify with certainty a local minimum. Additionally, Figure 2 presents a statistical comparison between the used optimization methods and the data provided from Table 3.



(a) With outliers appearing



(b) Without outliers appearing

Figure 2. Comparison using the number of function calls. The test was performed for different optimization methods.

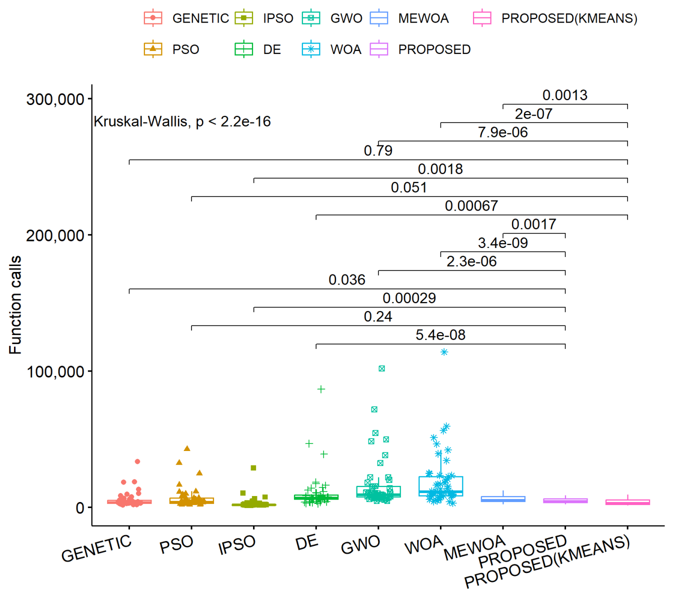


Figure 3. Statistical comparison of proposed method against others. The test was performed for different optimization methods

The table 3 contains values representing objective function evaluations, with lower values indicating better performance. The goal is to minimize the number of function calls required to achieve a result. The algorithms analyzed are GENETIC, PSO, IPSO, DE, GWO, WOA, MEWOA, PROPOSED, and PROPOSED(KMEANS). The GENETIC algorithm, with a total of 261,106 evaluations, shows mixed results. In some functions, such as SHEKEL10 with 2488 evaluations, it performs well. However, in more demanding functions, like DIFFPOWER5 and DIFFPOWER10 with 18,552 and 18,801 evaluations respectively, it requires many calls, indicating reduced efficiency. The PSO algorithm, with 317,168 total evaluations, needs more calls than GENETIC and is generally considered less efficient. In difficult functions, such as DIFFPOWER10, it requires many evaluations, indicating reduced effectiveness. The IPSO algorithm is the most efficient, with only 145,908 evaluations. Its low values in several functions, such as ROSENBROCK16 with

2432 calls and TEST2N5 with 1982 calls, indicate that it requires fewer evaluations to achieve a result, making it the most efficient algorithm. The DE algorithm, with 506,409 evaluations, performs worse compared to IPSO and other algorithms. In difficult functions, such as DIFFPOWER10 with 46,914 calls and SCHWEFEL222 with 86,861 calls, it requires significantly more evaluations, indicating difficulties in optimizing efficiently. GWO has the most total evaluations (802,608), showing reduced efficiency. It encounters particular difficulties in hard problems, such as DIFFPOWER10 with 54,442 calls and SCHWEFEL222 with 101,960 calls. WOA, with 916,996 evaluations, is even less efficient than GWO. It faces serious issues in demanding functions such as DIFFPOWER10 (56,381 calls) and SCHWEFEL222 (114,161 calls), indicating poor performance. MEWOA (Modified WOA) has 419,939 total evaluations, showing some improvement over WOA but still less efficient than IPSO and PROPOSED+KMEANS. Despite the improvement, it continues to require many calls in difficult problems, such as DIFFPOWER10 (38,559 calls) and SCHWEFEL222 (65,835 calls). The PROPOSED algorithm, with 361,454 evaluations, performs better than DE but does not reach the efficiency of IPSO. It shows moderate results in some problems, such as DIFFPOWER10 with 29,549 calls, limiting its overall performance. The improved PROPOSED(KMEANS) algorithm achieves better results, with 257,033 total evaluations. In many functions, such as SCHWEFEL222 with 22,058 evaluations and DIFFPOWER10 with 26,615 evaluations, the addition of KMEANS significantly improves performance, making it competitive with IPSO. In conclusion, IPSO is the most efficient algorithm, with the lowest total number of evaluations, achieving better results with fewer calls. PROPOSED(KMEANS) also performs well, approaching the performance of IPSO. On the other hand, GWO and WOA have the worst performance, requiring the highest number of evaluations. MEWOA shows some improvements over WOA but remains less efficient than the top-performing algorithms.

In Figure 3, pairwise comparisons between the proposed methods and the other methods are presented. In all these comparisons, the critical parameter $p<0.05$ confirms the statistical significance of the results.

Overall, the proposed methods achieve a significant reduction in the required number of objective function calls and, in many cases, outperform other optimization techniques. Compared to the Differential Evolution (DE) algorithm, the proposed approaches perform better across all mathematical models, with the reduction in the number of calls exceeding 50%. This trend is confirmed by Table 3. Additionally, Figures 2a, 2b, and 3 present statistical comparisons between the global optimization methods used in the experiments, highlighting the superiority of the proposed approaches.

The table 4 presents the number of objective function calls for different values of the differential weight F. When F is randomly selected within the range [0.0, 1.0], the total number of calls is 361,454. With F=0.5, the calls increase to 366,966, while with F=0.9, they decrease to 342,529, indicating the best performance. This suggests that higher values of F improve convergence by reducing the total function calls. Although random F values provide a good balance between exploration and exploitation, they do not outperform the fixed value F=0.9. On the other hand, F=0.5 appears to burden the algorithm with more function calls. Therefore, selecting a higher or adaptive F can enhance the algorithm's efficiency by lowering computational costs.

From the data in table 5, we observe that as the population size decreases, the number of function calls generally decreases as well, indicating increased efficiency with smaller populations. For example, the ACKLAY function requires 5,818 calls with a population of 200, 4,647 with a population of 150, and 2,829 with a population of 100. A similar trend is seen with DIFFPOWER10, which reduces from 29,549 to 22,112 and then to 15,325 as the population size decreases. However, there is variation between functions. SCHWEFEL222 shows very high call numbers, particularly with a population of 200 (78,734 calls), suggesting increased complexity. In contrast, functions like GKLS350 require fewer calls, even with larger populations. Although calls often decrease with a smaller population, this does not necessarily mean that the solution quality is maintained. Choosing the right population size is crucial, as excessive reduction may limit the search space and degrade optimization performance. Table 5 illustrates how population size impacts performance. Balancing population size and the number of calls is essential for achieving efficient and reliable results.

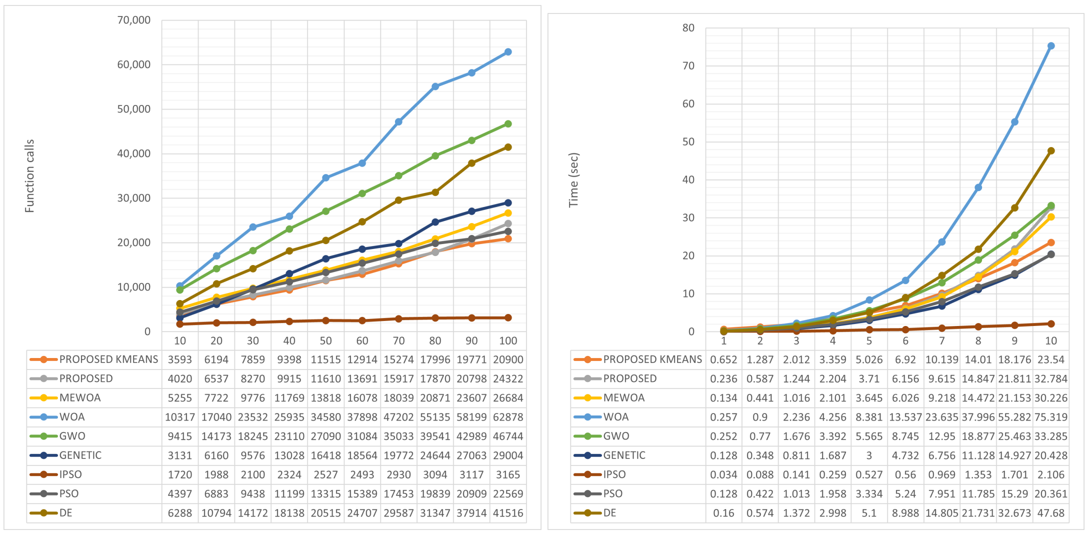
An additional experiment was executed for the High Elliptic function, which is defined as:

$$f(x) = \sum_{i=1}^{n} \left(10^6\right)^{\frac{i-1}{n-1}} x_i^2$$

where the parameter n defines the dimension of the function. In this particular experiment, the proposed optimization method was systematically applied to a specific mathematical function as the dimension n underwent a transition from 10 to 100. Figure 4a presents the number of objective function calls required for each algorithm across different dimensions. As the problem dimension increases, the number of function calls needed to find a solution grows significantly for all algorithms, indicating the increasing complexity of the ELP function. For example, the PROPOSED(KMEANS) algorithm requires 3593 calls for 10 dimensions and 20,900 calls for 100 dimensions, showing a gradual increase. Similarly, the PROPOSED algorithm starts with 4020 calls for 10 dimensions and reaches 24,322 calls for 100 dimensions. IPSO stands out as the most efficient algorithm, with only 1720 calls at 10 dimensions and 3165 at 100, consistently maintaining low evaluations. In contrast, WOA exhibits the worst performance, with 10,317 calls at 10 dimensions and 62,878 at 100 dimensions. The rate of increase in calls varies among algorithms. For example, the PROPOSED algorithm increases from 4020 to 6537 calls between 10 and 20 dimensions (a 1.6 times increase), while the rise from 90 to 100 dimensions is more moderate, from 20,798 to 24,322 calls. MEWOA, although improved compared to WOA, requires more calls than the proposed methods, with 5255 calls at 10 dimensions and 26,684 at 100. GENETIC displays a variable pattern, with 3131 calls at 10 dimensions and 29,004 at 100. GWO and PSO follow a similar trend, with a gradual increase in calls as the dimension increases. Additionally, Figure 4b illustrates the execution time for each algorithm across different dimensions, measured in seconds. As expected, increasing the dimension leads to a significant rise in execution time for all algorithms, highlighting the increased computational load. For example, PROPOSED(KMEANS) starts with 0.652 seconds at 10 dimensions and reaches 23.54 seconds at 100. PROPOSED begins with 0.236 seconds at 10 dimensions and climbs to 32.784 seconds at 100, marking an impressive 139-fold increase. The fastest algorithm, IPSO, takes only 0.034 seconds at 10 dimensions and 2.106 seconds at 100, demonstrating excellent performance and scalability. On the other hand, WOA records the worst performance, with 75.319 seconds at 100 dimensions compared to 0.257 seconds at 10 dimensions. DE also struggles in execution time, increasing from 0.16 seconds at 10 dimensions to 47.68 seconds at 100. The increase in execution time across dimensions follows a nonlinear trend for most algorithms. For example, PROPOSED takes 0.587 seconds at 20 dimensions (more than double the 0.236 seconds at 10 dimensions) but reaches 14.847 seconds at 80 dimensions. Similarly, PSO starts at 0.128 seconds at 10 dimensions and reaches 20.361 seconds at 100 dimensions.

The data reveals significant differences in the scalability and efficiency of the algorithms. IPSO stands out as the most efficient algorithm, with the lowest number of calls and the shortest execution time. PROPOSED(KMEANS) also achieves satisfactory results, maintaining a balance between calls and execution time. On the other hand, WOA and GWO exhibit the worst performance, with significantly more calls and substantial increases in execution time as the dimension grows.

The results confirm that the complexity of optimization problems increases sharply with the dimension. Most algorithms struggle to maintain their efficiency in higher dimensions, highlighting the importance of selecting the appropriate algorithm according to the problem's requirements and the available computational resources.



	10	20	30	40	50	60	70	80	90	100
PROPOSED KMEANS	3593	6194	7859	9398	11515	12914	15274	17996	19771	20900
PROPOSED	4020	6537	8270	9915	11610	13691	15917	17870	20798	24322
MEWOA	5255	7722	9776	11769	13818	16078	18039	20871	23607	26684
WOA	10317	17040	23532	25935	34580	37898	47202	55135	58199	62878
GWO	9415	14173	18245	23110	27090	31084	35033	39541	42989	46744
GENETIC	3131	6160	9576	13028	16418	18564	19772	24644	27063	29004
IPSO	1720	1988	2100	2324	2527	2493	2930	3094	3117	3165
PSO	4397	6883	9438	11199	13315	15389	17453	19839	20909	22569
DE	6288	10794	14172	18138	20515	24707	29587	31347	37914	41516

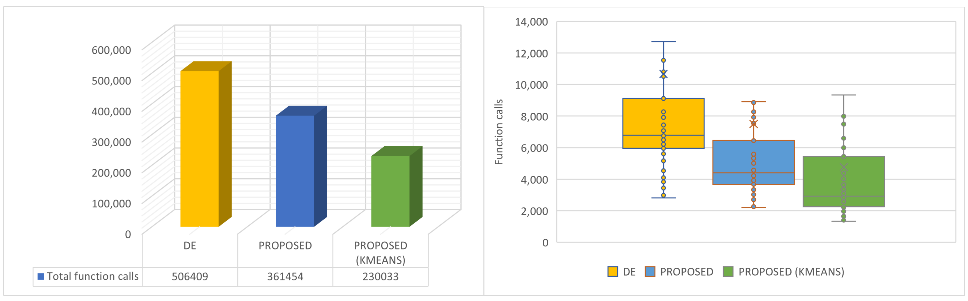
	1	2	3	4	5	6	7	8	9	10
PROPOSED KMEANS	0.652	1.287	2.012	3.359	5.026	6.92	10.139	14.01	18.176	23.54
PROPOSED	0.236	0.587	1.244	2.204	3.71	6.156	9.615	14.847	21.811	32.784
MEWOA	0.134	0.441	1.016	2.101	3.645	6.026	9.218	14.472	21.153	30.226
WOA	0.257	0.9	2.236	4.256	8.381	13.537	23.635	37.996	55.282	75.319
GWO	0.252	0.77	1.676	3.392	5.565	8.745	12.95	18.877	25.463	33.285
GENETIC	0.128	0.348	0.811	1.687	3	4.732	6.756	11.128	14.927	20.428
IPSO	0.034	0.088	0.141	0.259	0.527	0.56	0.969	1.353	1.701	2.106
PSO	0.128	0.422	1.013	1.958	3.334	5.24	7.951	11.785	15.29	20.361
DE	0.16	0.574	1.372	2.998	5.1	8.988	14.805	21.731	32.673	47.68

(a) Comparison of the function calls **(b)** Time comparison

Figure 4. Different variations of the ELP function of the proposed method with other methods

342

In each iteration of the algorithm, a trial point is calculated through vector operations, similar to the process of optimization using differential evolution. The main difference between the proposed method compared to differential evolution lies in the fact that the samples for calculating the trial point are selected from nearby regions of the initial distribution, rather than being chosen randomly. However, the performance of the two methods differs in their ability to find optimal solutions, as shown in Figure 1. In the case where the initial distribution is created by the k-means algorithm, the performance increases by 36.3%. The statistical analysis of the results in Figure 3 and Figure 5 indicates that the proposed methods (PROPOSED and PROPOSED+KMEANS) demonstrate statistically significantly better performance than the DE method in most cases. The use of the t-test confirms that these differences are not random but are due to substantial improvements in the efficiency of the proposed methods. The lower function call values show that the proposed methods lead to faster convergence, and thus, better optimization performance. The extraneous samples of Figure 5b have been removed.



	DE	PROPOSED	PROPOSED (KMEANS)
■ Total function calls	506409	361454	230033

(a) Comparison of total function calls **(b)** Statistical comparison of total function calls

Figure 5. Performance comparison of proposed methods against differential evolution: Effect of K-Means algorithm on convergence.

357

4. Conclusions

An innovative global optimization method has been proposed in this research paper, which leverages techniques derived from well-established optimization strategies. More specifically, the new method incorporates genetic operators from Genetic Algorithms alongside the Linear Search method to generate candidate solutions for the given objective functions. These candidate solutions are then combined to create new solutions utilizing approaches inspired by the Differential Evolution method. To validate the effectiveness of this new optimization approach, a comprehensive series of experiments were conducted on various problems sourced from the existing literature. Additionally, numerical comparisons were made with recognized global optimization techniques to provide a clear benchmark. The results indicate that the proposed optimization method exhibits significantly superior performance when compared to alternative methods, particularly in terms of the number of calls made to the objective function. Fewer calls to the objective function suggest better overall efficiency, highlighting the proposed method's ability to achieve optimal solutions with fewer evaluations. This efficiency is particularly critical in scenarios where each function call is computationally expensive. Statistical analysis, including both the t-test and the Kruskal-Wallis test, confirm that the observed differences in the number of calls between the proposed method and other methods are statistically significant, with a p-value of less than 0.05. This finding not only underscores the reduced resource consumption of the proposed method but also affirms that it delivers reliable results with enhanced efficiency. In summary, the proposed method stands out in terms of efficiency when compared to other optimization techniques, significantly decreasing the number of objective function calls and optimizing overall computational cost. Potential enhancements to the algorithm could involve identifying specific samples that contribute more effectively to the discovery of the optimal solution. Furthermore, since this method represents a novel approach to optimization, exploring alternative termination criteria or varying the initial sample distributions could lead to even greater performance improvements. By refining these aspects, the method could further bolster its efficiency and effectiveness in solving complex optimization problems.

Author Contributions: The conceptualization of the idea and the design of the methodology, as well as the supervision of the technical aspects related to the software, were undertaken by V.C., I.G.T., and G.K. The experiments were conducted using various datasets, and the comparative results were presented by V.C., G.K. and A.M.G. The statistical analysis was carried out by V.C. The manuscript was prepared by G.K. and the other authors. All authors have reviewed and approved the final published version.

Funding: No external funding was received for this research.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Institutional Review Board Statement: Not applicable.

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.

Conflicts of Interest: No conflicts of interest are declared by the authors.

References

1. Carrizosa, E., Molero-Río, C., & Romero Morales, D. (2021). Mathematical optimization in classification and regression trees. Top, 29(1), 5-33.
2. Stilck França, D., & Garcia-Patron, R. (2021). Limitations of optimization algorithms on noisy quantum devices. Nature Physics, 17(11), 1221-1227.

3. Zhang, J., & Glezakou, V. A. (2021). Global optimization of chemical cluster structures: Methods, applications, and challenges. International Journal of Quantum Chemistry, 121(7), e26553.
4. 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.
5. Hesami, M., & Jones, A. M. P. (2020). Application of artificial intelligence models and optimization algorithms in plant cell and tissue culture. Applied Microbiology and Biotechnology, 104(22), 9449-9485.
6. Filip, M., Zoubek, T., Bumbalek, R., Cerny, P., Batista, C. E., Olsan, P., ... & Findura, P. (2020). Advanced computational methods for agriculture machinery movement optimization with applications in sugarcane production. Agriculture, 10(10), 434.
7. Alirahmi, S. M., Mousavi, S. B., Razmi, A. R., & Ahmadi, P. (2021). A comprehensive techno-economic analysis and multi-criteria optimization of a compressed air energy storage (CAES) hybridized with solar and desalination units. Energy Conversion and Management, 236, 114053.
8. Shezan, S. A., Ishraque, M. F., Shafiullah, G. M., Kamwa, I., Paul, L. C., Muyeen, S. M., ... & Kumar, P. P. (2023). Optimization and control of solar-wind islanded hybrid microgrid by using heuristic and deterministic optimization algorithms and fuzzy logic controller. Energy reports, 10, 3272-3288.
9. Hsieh, Y. P., Karimi Jaghargh, M. R., Krause, A., & Mertikopoulos, P. (2024). Riemannian stochastic optimization methods avoid strict saddle points. Advances in Neural Information Processing Systems, 36.
10. Villanueva, F. R., & de Oliveira, V. A. (2022). Necessary optimality conditions for interval optimization problems with functional and abstract constraints. Journal of Optimization Theory and Applications, 194(3), 896-923.
11. 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.
12. Charilogis, V., Tsoulos, I. G., & Stavrou, V. N. (2023). An Intelligent Technique for Initial Distribution of Genetic Algorithms. Axioms, 12(10), 980.
13. Deng, W., Shang, S., Cai, X., Zhao, H., Song, Y., & Xu, J. (2021). An improved differential evolution algorithm and its application in optimization problem. Soft Computing, 25, 5277-5298.
14. Shami, T. M., El-Saleh, A. A., Alswaitti, M., Al-Tashi, Q., Summakieh, M. A., & Mirjalili, S. (2022). Particle swarm optimization: A comprehensive survey. Ieee Access, 10, 10031-10061.
15. Rokbani, N., Kumar, R., Abraham, A., Alimi, A. M., Long, H. V., Priyadarshini, I., & Son, L. H. (2021). Bi-heuristic ant colony optimization-based approaches for traveling salesman problem. Soft Computing, 25, 3775-3794.
16. Pourpanah, F., Wang, R., Lim, C. P., Wang, X. Z., & Yazdani, D. (2023). A review of artificial fish swarm algorithms: Recent advances and applications. Artificial Intelligence Review, 56(3), 1867-1903.
17. Kareem, S. W., Mohammed, A. S., & Khoshabaa, F. S. (2023). Novel nature-inspired meta-heuristic optimization algorithm based on hybrid dolphin and sparrow optimization. International Journal of Nonlinear Analysis and Applications, 14(1), 355-373.
18. Nadimi-Shahraki, M. H., Zamani, H., Asghari Varzaneh, Z., & Mirjalili, S. (2023). A systematic review of the whale optimization algorithm: theoretical foundation, improvements, and hybridizations. Archives of Computational Methods in Engineering, 30(7), 4113-4159.
19. Lalwani, S., Sharma, H., Satapathy, S. C., Deep, K., & Bansal, J. C. (2019). A survey on parallel particle swarm optimization algorithms. Arabian Journal for Science and Engineering, 44, 2899-2923.
20. Hussain, M. M., & Fujimoto, N. (2020). GPU-based parallel multi-objective particle swarm optimization for large swarms and high dimensional problems. Parallel Computing, 92, 102589.
21. Eshelman, L. J. (2018). Genetic algorithms. In Evolutionary Computation 1 (pp. 102-118). CRC Press.
22. Hervis Santana, Y., Martinez Alonso, R., Guillen Nieto, G., Martens, L., Joseph, W., & Plets, D. (2022). Indoor genetic algorithm-based 5G network planning using a machine learning model for path loss estimation. Applied Sciences, 12(8), 3923.
23. Liu, X., Jiang, D., Tao, B., Jiang, G., Sun, Y., Kong, J., ... & Chen, B. (2022). Genetic algorithm-based trajectory optimization for digital twin robots. Frontiers in Bioengineering and Biotechnology, 9, 793782.

24. Liu, K., Deng, B., Shen, Q., Yang, J., & Li, Y. (2022). Optimization based on genetic algorithms on energy conservation potential of a high speed SI engine fueled with butanol–gasoline blends. Energy Reports, 8, 69-80.
25. MirRokni, M. K. S. M. (2017). Applying genetic algorithm in architecture and neural network training. International Journal of Computer Science and Network Security IJCSN, 17(6), 118.
26. Li, Y. H., Wang, J. Q., Wang, X. J., Zhao, Y. L., Lu, X. H., & Liu, D. L. (2017). Community detection based on differential evolution using social spider optimization. Symmetry, 9(9), 183.
27. Maulik, U., & Saha, I. (2010). Automatic fuzzy clustering using modified differential evolution for image classification. IEEE transactions on Geoscience and Remote sensing, 48(9), 3503-3510.
28. Vivekanandan, T., & Iyengar, N. C. S. N. (2017). Optimal feature selection using a modified differential evolution algorithm and its effectiveness for prediction of heart disease. Computers in biology and medicine, 90, 125-136.
29. Wu, T., Li, X., Zhou, D., Li, N., & Shi, J. (2021). Differential evolution based layer-wise weight pruning for compressing deep neural networks. Sensors, 21(3), 880.
30. Badem, H., Basturk, A., Caliskan, A., & Yuksel, M. E. (2018). A new hybrid optimization method combining artificial bee colony and limited-memory BFGS algorithms for efficient numerical optimization. Applied Soft Computing, 70, 826-844.
31. Li, S., Tan, M., Tsang, I. W., & Kwok, J. T. Y. (2011). A hybrid PSO-BFGS strategy for global optimization of multimodal functions. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 41(4), 1003-1014.
32. Andalib Sahnehsaraei, M., Mahmoodabadi, M. J., Taherkhorsandi, M., Castillo-Villar, K. K., & Mortazavi Yazdi, S. M. (2015). A hybrid global optimization algorithm: particle swarm optimization in association with a genetic algorithm. Complex System Modelling and Control Through Intelligent Soft Computations, 45-86.
33. Huang, P., Huang, H. Z., & Huang, T. (2019). A novel algorithm for structural reliability analysis based on finite step length and Armijo line search. Applied Sciences, 9(12), 2546.
34. Charilogis, V., & Tsoulos, I. G. (2022). Toward an ideal particle swarm optimizer for multidimensional functions. Information, 13(5), 217.
35. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in engineering software, 69, 46-61.
36. Ahmed, M., Seraj, R., & Islam, S. M. S. (2020). The k-means algorithm: A comprehensive survey and performance evaluation. Electronics, 9(8), 1295.
37. Dai, Y. H. (2002). Convergence properties of the BFGS algoritm. SIAM Journal on Optimization, 13(3), 693-701.
38. 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.
39. 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.
40. Zabinsky, Z. B., Graesser, D. L., Tuttle, M. E., & Kim, G. I. (1992). Global optimization of composite laminates using improving hit and run. In Recent advances in global optimization (pp. 343-368).
41. 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.
42. 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.
43. Storn, R. (1996, June). On the usage of differential evolution for function optimization. In Proceedings of North American fuzzy information processing (pp. 519-523). Ieee.
44. Shen, Y., Zhang, C., Gharehchopogh, F. S., & Mirjalili, S. (2023). An improved whale optimization algorithm based on multi-population evolution for global optimization and engineering design problems. Expert Systems with Applications, 215, 119269**.**
45. Powell, M. J. D. (1989). A tolerant algorithm for linearly constrained optimization calculations. Mathematical Programming, 45, 547-566.