Article

A novel method that is based on Differential Evolution suitable for large scale optimization problems

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

Abstract

Global optimization represents a fundamental challenge in computer science and engineering, as it aims to identify high-quality solutions to problems spanning from moderate to extremely high dimensionality. The DE algorithm is a population-based algorithm like genetic algorithms and uses similar operators such as: crossover, mutation and selection. The proposed method introduces a set of methodological enhancements designed to increase both the robustness and the computational efficiency of the classical DE framework. Specifically, an adaptive termination criterion is incorporated, enabling early stopping based on statistical measures of convergence and population stagnation. Furthermore, a population sampling strategy based on k-means clustering is employed to enhance exploration and improve the redistribution of individuals in high-dimensional search spaces. This mechanism enables structured population renewal and effectively mitigates premature convergence. The enhanced algorithm was evaluated on standard large-scale numerical optimization benchmarks and compared with established global optimization methods. The experimental results indicate substantial improvements in convergence speed, scalability, and solution stability.

Keywords: Optimization; Differential Evolution; Evolutionary techniques; Stochastic methods; Large-scale problems

Received:
Revised:
Accepted:
Published:

Citation: . A novel method that is based on Differential Evolution suitable for large scale optimization problems. *Journal Not Specified* **2025**, *1*, 0. https://doi.org/

1. Introduction

The basic goal of global optimization is to find the global minimum of a continuous multidimensional function and is defined as:

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

with S:

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

with, a_i and b_i representing lower and upper bounds for each variable x_i.

In recent years many researchers have published important reviews on global optimization. Such methods find application in a wide range of scientific fields, such as mathematics [1,2], physics [3,4], chemistry [5,6], biology [9,10], medicine [7,8], agriculture [11,12] and economics [13,14]. A particular challenge is the large-scale global optimization

(LSGO) problem, where the complexity increases significantly with increasing problem dimensions. Finding efficient and computationally feasible solutions has become particularly difficult, which has led the research community to focus on the development of innovative algorithms. LSGO problems are encountered in a wide range of applications, while their importance is also reflected in the organization of the first global large-scale optimization competition within the framework of the CEC in 2008. Other competitions followed in 2010 [15], 2013 [16] and 2015 [17], attracting the intense interest of the academic community.

To address these challenges, various heuristic and meta-heuristic approaches have been developed. Evolutionary Algorithms (EA) [18,19]are one of the most effective categories, as they mimic natural selection and genetic evolution to search for best solutions. Due to their adaptability and robustness, EAs can solve difficult optimization problems. Some of the most well-known EAs are Differential Evolution (DE) [20,21], Genetic Algorithms [22,23], Evolutionary Strategies [24,25], Evolutionary Programming[26,27] , Multimodal Optimization Algorithms [28,29]. Also, methods inspired by Swarm Intelligence [30,31] such as: Particle Swarm Optimization (PSO) [32,33], Ant Colony Optimization (ACO) [34,35], Artificial Bee Colony Optimization (ABC) [36,37], Firefly Algorithm (FA) [38,39] , Bat Algorithm [40,41] are strong alternatives.

Differential Evolution (DE) is one of the most widely used optimization techniques, as it offers high robustness, simplicity and fast convergence. DE is a highly efficient evolutionary algorithm that has gained significant recognition since the late 1990s. DE, originally introduced in 1995 by Storn and Price [42,43], has proven to be a versatile optimization tool that can be applied in various scientific and engineering fields. It is particularly effective for symmetric optimization problems, as well as for dealing with discontinuous, noisy, and dynamic challenges. In physics, it has been used in energy-related problems, including wind power optimization. In chemistry, it has contributed to advances in atmospheric chemistry [44]and the development of high-performance chemical reactors. [45] DE has also had significant implications in health-related areas, such as breast cancer research [46] and medical diagnostics. [47]Despite its effectiveness, DE has some limitations, such as the difficulty of adapting its parameters to different problems and its reduced performance in high-dimensional environments. Furthermore, while it has a powerful exploration mechanism, it often lags in exploiting the identified solutions, which can slow down the convergence process. To address these challenges, several improved versions of DE have been proposed. The goal of these improvements is to achieve a better balance between exploration and exploitation, parameter adaptability, and more effectively handle large-scale problems. The continuous research activity in this field demonstrates the importance of LSGO and the need to develop increasingly efficient algorithms to deal with it.

The current work introduces a number of modifications to the DE algorithm in order to speed up the process and increase the efficiency of the algorithm, especially for large - scale problems. The modifications introduced in DE aim to improve performance on large-scale problems. In more detail:

- Sampling Method: The use of k-means for sampling contributes to improving the quality of the initial solutions and reducing the dimensional complexity, leading to faster convergence.
- Termination Technique for Differential Evolution: The proposed technique allows for early termination of the process when no significant improvement is observed, thus reducing the number of unnecessary evaluations of the objective function.
- Different mechanisms for the differential weighted parameter: The integration of Number, Random and Migrant approaches allows greater adaptability of the algorithm to the requirements of each problem.

- Periodic Local Optimization Refinement: The use of local methods, such as BFGS, contributes to further improving the solutions, increasing the accuracy of the final results.

The combined use of these techniques enhances the efficiency and reliability of DE in large-scale problems, offering a more efficient way of searching for solutions.

Recent studies have proposed various strategies to address large-scale optimization challenges, including cooperative coevolution [50], Particle Swarm Optimization [51], a memetic DE approach [52], a self-adaptive Fast Fireworks Algorithm[53], swarm-based methods with learning mechanisms[77],[78] and advanced decomposition techniques such as dual Differential Grouping [56].

The remains of this paper are divided as follows: in section 2 the original DE algorithm, the proposed method as well as the flowchart with detailed description are presented, in section 3 of the test functions used in the experiments as well as the related experiments are presented. In the 4 section, there is a brief discussion of the results obtained from the experiments. In section 5 some conclusions and directions for future improvements are discussed.

2. Materials and Methods

2.1 The original Differential Evolution method

1. **Set** the population size $NP \geq 4$,usually $NP = 10n$, where n is the dimension of the input problem.
2. **Create** randomly from a distribution NP agents $x_i,\ i = 1, \ldots NP$
3. **Set** the crossover probability $CR \in [0,1]$. A typical value for this parameter is 0.9.
4. **Set** the differential weight $F \in [0,2]$. A typical value for this parameter is 0.8.
5. **Initialize** all members of the population in the search space. The members of the population are called agents.
6. **Until** some stopping criterion is met, repeat:

(a) **For** $i = 1...NP$ **do**.

- **Set** x_i as the agent i.
- **Pick** randomly three agents a, b, c.
- **Pick** a random index $R \in 1, ..., n$.
- **Compute** the trial vector $y = [y_1, y_2, ..., y_n]$ as follows.
- **For** $j = 1, ..., n$ **do**:
  A. **Set** $r_i \in [0,1]$ a random number.
  B. **If** $r_j < CR\ or\ j = R$ then $y_j = a_j + F \times b_j - c_j\ else\ y_j = x_{ij}$.
- **If** $f(y) \leq f(x_i)$ then $x_i = y$.
- **EndFor.**

(b) **EndFor.**

6. **Return** the agent x_{best} in the population with the lower function value x_{best}.

The DE process begins by defining the population size NP, typically set as $NP = 10n$, where n is the problem's dimensionality. We define the crossover probability CR with a default value of 0.9 and the differential weight F with a default value of 0.8. We randomly initialize all members of the population, referred to as agents, within the search space. The method iterates until a termination criterion is met. In each iteration, for every agent x_i in the population, we randomly select three distinct agents a, b, and **c**. We then choose a random index R from 1 to n. Next, we construct a trial vector y by computing, for each component j, the value $y_j = a_j + F \times (b_j - c_j)$ if a random number r_j is less than CR or if $j = R$; otherwise, we keep $y_j = x_j$. If the objective function value $f(y)$ is better than or equal to $f(x_i)$, we replace x_i with y. At the end of the process, we return the best-performing agent x_{best}, which has the optimal objective function value. The method combines stochastic search with directional variations derived from differences between population agents, ensuring an efficient exploration-exploitation balance in the search space.

2.2 The proposed Differential Evolution method

The proposed algorithm incorporates a series of modifications to the Original DE method, which makes finding the global minimum in high - dimensional problems more efficient. The main steps of the proposed method are listed subsequently.

1. **Initialization step**.
   (a) **Set** as NP the population size of the method (number of agents).
   (b) **Create** randomly from a distribution NP agents $x_i,\ i = 1, \ldots NP$
   (c) **Compute** the fitness value f_i of each agent x_i using the objective function as $f_i = f(x_i)$.
   (d) **Set** as p_l the local search rate.
   (e) **Set** the integer parameter N_t as the tournament size.
   (f) **Set** as N_g the maximum number of iterations allowed.
   (g) **Set** as N_I the number of iterations used in the stopping rule.
   (h) **Set** $k = 0$, the iteration counter.
   (i) **Set** the parameter CR, which represents the crossover probability with $CR \leq 1$.
   (j) **Select** the differential weight method, which is represented by the parameter F. In the proposed method three distinct methods were incorporated:
      i. **Number**. In this case the parameter F is chosen as a constant value.
      ii. **Random**. The random method represents the differential weight mechanism proposed by Charilogis et al. [58], where it is defined as:
      $$F = -0.5 + 2r \tag{2}$$
      where r is a random number with $r \in [0, 1]$.
      iii. **Migrant**. In this case the differential weight mechanism proposed in [59] was used.
2. **For** $i = 1, \ldots, NP$ **do**
   (a) **Select** the agent x_i
   (b) **Select** randomly three distinct agents x_a, x_b, x_c. The selection of these agents could be performed randomly or with the application of the tournament selection procedure. During tournament selection, a subset of N_t agents are selected from the current population and the one with the lowest fitness value is selected.
   (c) **Choose** a random integer $R \in [1, n]$,where n is the dimension of the objective problem.
   (d) **Create** a trial point x_t.
   (e) **For** $j = 1, \ldots .n$ **do**
      i. **Select** a random number $r \in [0, 1]$.
      ii. **If** $r \leq CR$ **or** $i = R$ **then** $x_{t,j} = x_{a,j} + F \times \left(x_{b,j} - x_{c,j}\right)$ **else** $x_{t,j} = x_{i,j}$
   (f) **End For**
   (g) **Set** $y_t = f(x_t)$
   (h) **If** $y_t \leq f(x_i)$ **then** $x_i = x_t,\ f(x_i) = y_t$.
   (i) **Select** a random number $r \in [0, 1]$. If $r \leq p_l$ then $x_i = \mathrm{LS}(x_i)$,where LS defines a local search procedure. In the proposed method the BFGS variant of Powell [60] was used.

3. **End For**
4. **Check for termination**.
   (a) **Set** $k = k + 1$
   (b) **If** $k \geq N_g$ then terminate.
   (c) **Check** the termination rule specified in the work of Charilogis et al [58]. In this work the quantity

$$\delta^{(k)} = \left| \sum_{i=1}^{\mathrm{NP}} \left| f_i^{(k)} \right| - \sum_{i=1}^{\mathrm{NP}} \left| f_i^{(k-1)} \right| \right| \quad (3)$$

   is calculated. The term $f_i^{(k)}$ stands for the fitness value of agent i at iteration k. If $\delta^{(k)} \leq \epsilon$ for a number of N_I iterations, then terminate the algorithm else goto step 2.

The modified Differential Evolution method begins with the initialization step. We define the population size NP as the number of agents. We randomly create NP agents x_i, $i = 1$ to NP, from some distribution and calculate the fitness value $f_i = f(x_i)$ for each agent using the objective function. We define the local search rate p_l, the tournament size N_t, the maximum number of iterations N_g, the number of iterations N_I for the termination rule, and set the iteration counter $k = 0$. We define the crossover probability CR and select the method for the differential weight F, which can be constant, random (F = -0.5 + 2r), where r is a random number in [0,1]), or based on the Migrant method.

Next, for each agentx_i, $i = 1$ to NP, we randomly select three distinct agents x_a, x_b, x_c, either through random selection or through a tournament procedure where we select a subset of N_t agents and choose the one with the lowest fitness value. We select a random integer R in [1,n], where n is the problem's dimensionality, and create a trial point x_t. For each dimension $j = 1$ to n, we select a random number r in [0,1]. If $r \leq CR$ or$j = R$, then $x_{t,j} = x_{a,j} + F \times \left(x_{b,j} - x_{c,j} \right)$, otherwise$x_t, j = x_i, j$. We calculate the fitness value$y_t = f(x_t)$ and if$y_t \leq f(x_i)$, we replace x_i with x_t and $f(x_i)_i$ with y_t. We select a random number r in [0,1], and if $r \leq p_l$, we apply a local search $LS(x_i)$ to x_i using Powell's BFGS method.

After completing the iteration, we update the counter $k = k + 1$. If $k \geq N_g$, we terminate the algorithm. Otherwise, we calculate the quantity 3, where $f_i^{(k)}$ is the fitness value of agent i at iteration k. If $\delta^{(k)} \leq \varepsilon$ for N_I iterations, we terminate the algorithm otherwise, we continue with the next iteration. This method incorporates random variations, local search, and different mechanisms for the differential weight F, aiming to improve search efficiency.

2.3 The Flowchart of the proposed Differential Evolution method

The steps of the proposed DE algorithm can be described as follows:

The process begins by initializing the basic parameters, including population size, differential weight values, and crossover probabilities. The initial population is then generated, either uniformly distributed over the search space or clustered using the k-means algorithm. Once the initial population is determined, the fitness of each individual is evaluated and stored in a fitness table. At this stage, a differential weight calculation method is selected, with Stable, Random or Migrant options. Likewise, the sampling procedure for generating new solutions is specified, which may include tournament selection or random sampling. To further refine the candidate solutions, a local search procedure is applied, improving the quality of the trial solutions. The algorithm then checks the termination criteria to decide whether to proceed or terminate. If the criteria are still not met, the process returns to update the fitness table and continues to repeat. If the termination criteria are met, the algorithm outputs the best solution found.

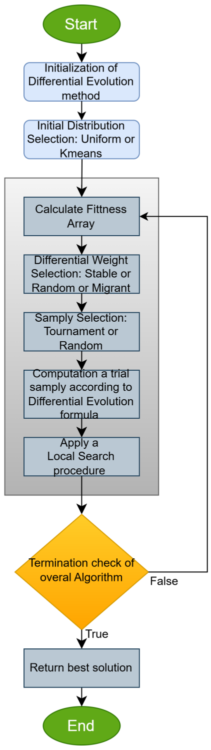

Figure 1. The steps of the proposed DE algorithm.

3. Experiments

This section begins with a description of the functions that will be used in the experiments and then presents in detail the experiments that were performed, in which the parameters available in the proposed algorithm were studied, in order to study their reliability and adequacy.

3.1. Test Functions

A variety of test functions were used in the conducted experiments. These functions are used in a series of research papers [67–70]. In the present research work, these functions were used with a varying number of dimensions from 25 to 150. The description of each used test function is provided below. In all cases, the constant n defines the dimension of the objective function.

3.2. Experimental results

A series of experiments were carried out for the previously mentioned functions and these experiments were executed on an AMD RYZEN 5950X with 128GB RAM. The operating system of the running machine was Debian Linux. Each experiment was conducted 30 times, with different random numbers each time, and the averages were recorded. The software used in the experiments was coded in ANSI C++ using the freely available optimization environment of OPTIMUS[71], which can be downloaded from https://github.com/itsoulos/OPTIMUS. The values for the experimental parameters used in the proposed method are outlined in the Table 1

NAME	FORMULA	DIM	G_{min}
ATTRACTIVE SECTOR	$f(x)=\left(\sum_{i=1}^{n}(s_i x_i)^2\right)^{0.9}$	2	0
BUCHE RASTRIGIN	$f(\mathbf{x})=\sum_{i=1}^{n}\left[z_i\cdot(1+0.1\cdot\sin(10\pi z_i))\right]$	n	0
DIFFERENT POWERS	$f(\mathbf{x})=\sqrt{\sum_{i=1}^{n}\lvert x_i\rvert^{2+4\frac{i-1}{n-1}}}$	n	0
DISCUS	$f(x)=10^6 x_1^2+\sum_{i=2}^{n}x_i^2$	n	0
ELLIPSOIDAL	$f(x)=\sum_{i=1}^{n}\left(10^6\right)^{\frac{i-1}{n-1}}x_i^2$	n	0
GALLAGHER101	$f(\mathbf{x})=\max_{i=1}^{101}\left[h_i-w_i\sqrt{\sum_{j=1}^{n}(x_j-c_{ij})^2}\right]$ $min: 100+1$	n	0
GALLAGHER21	$f(\mathbf{x})=\max_{i=1}^{21}\left[h_i-w_i\sqrt{\sum_{j=1}^{n}(x_j-c_{ij})^2}\right]$ $min: 10+10+1$	n	0
GRIEWANK ROSENBROCK	$f(\mathbf{x})=\underbrace{\left(\frac{\lVert\mathbf{x}\rVert^2}{4000}-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1\right)}_{\text{Griewank}}\cdot\underbrace{\left(\frac{1}{10}\sum_{i=1}^{n-1}\left[100(x_{i+1}-x_i^2)^2+(1-x_i)^2\right]\right)}_{\text{Rosenbrock}}$	n	0
GRIEWANK	$f(x)=1+\frac{1}{200}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\frac{\cos(x_i)}{\sqrt{(i)}}$	n	0
RARSTIGIN	$f(\mathbf{x})=An+\sum_{i=1}^{n}\left[x_i^2-A\cos(2\pi x_i)\right]$ $A=10$	n	0
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	0
SHARP RIDGE	$f(\mathbf{x})=x_1^2+\alpha\sum_{i=2}^{n}x_i^2,\ a>1$	n	0
SPHERE	$f(\mathbf{x})=\sum_{i=1}^{n}x_i^2$	n	0
STEP ELLIPSOIDAL	$f(\mathbf{x})=\sum_{i=1}^{n}\lfloor x_i+0.5\rfloor^2+\alpha\sum_{i=1}^{n}\left(10^6\cdot\frac{i-1}{n-1}\right)x_i^2,\ a=1$	n	0
ZAKHAROV	$f(\mathbf{x})=\sum_{i=1}^{n}x_i^2+\left(\sum_{i=1}^{n}\frac{i}{2}x_i\right)^2+\left(\sum_{i=1}^{n}\frac{i}{2}x_i\right)^4$	n	0

Table 1. The values of the parameters of the proposed method.

PARAMETER	MEANING	VALUE
NP	Number of agents for all methods	200
n	Maximum number of allowed iterations for all methods	200
p_l	Local search rate,	0.05
F	Differential weight for classic DE	$F \in [0, 1.0]$
F	Differential weight for PROPOSED	0.8
CR	Crossover probability	0.9
N_I	Number of iterations used in the termination rule	8
-	Mutation rate for GA	0.05 (5%)
-	Selection Rate for GA	0.05 (5%)
-	Selection method fo GA	Roulette

In the following tables that depict the experimental results, the numbers in cells stand for the average function calls, as measured on 30 independent runs. The numbers in parentheses denote the fraction of the executions where the method successfully discovered the global minimum. If this number is not present, then the method managed to locate the global minimum in every run (100% success).

3.3. The proposed method in comparison with others

The Table 2 presents the results of a comparative analysis of various optimization methods (BICCA[72], MLSHADESPA[73], SHADE_ILS[74], Differential Evolution (DE)[20, 21], Genetic Algorithm (GA)[22,23], Whale Optimization Algorithm(WOA) [75,76], Particle Swarm Optimization (IPSO)[32,33], PROPOSED) across a wide range of test functions with dimensions of 25, 50, 100, and 150. Each row corresponds to a test function, while the columns represent the methods. The numerical values in each cell indicate the number of objective function calls required to find the minimum, while the values in parentheses show the success rate of each method in each case. In the last row (TOTAL), the total sum of function calls for each method is displayed, along with the average success rate. The best methods should simultaneously exhibit a low number of function calls (efficiency) and a high success rate (reliability). The analysis shows that the Proposed method delivers strong and consistent performance. Its overall success rate (0.85) is comparable to those of GA, MLSHADESPA, SHADE_ILS, IPSO, DE and WOA (around 0.90), and distinctly higher than the BICCA method (0.73). This indicates that the Proposed method remains dependable in locating the global minimum even when faced with complex or high-dimensional search spaces. In terms of computational cost, the Proposed method requires a total of

387,335 objective function evaluations, which is substantially lower than most competing techniques. This advantage appears consistently across the majority of tested functions. For example, in the DifferentPowers function, the Proposed method significantly outperforms GA across all dimensionalities: at 25 dimensions it uses 6,478 evaluations compared to 14,495 for GA at 100 dimensions, 16,225 versus 28,413 and at 150 dimensions, 21,495 versus 33,569. Similar observations are made for the GriewankRosenbrock function, where the Proposed method demonstrates clear efficiency benefits: at 100 dimensions it requires 6,465 evaluations whereas BICCA needs 20,462, and at 150 dimensions the gap widens further with 7,272 evaluations compared to 30,604 for BICCA. These differences illustrate the method's robustness and its ability to maintain low computational demands in highly nonlinear and difficult optimization landscapes.

In summary, the findings suggest that the Proposed method offers a strong balance between reliability and computational efficiency. It competes effectively with and in many cases surpasses widely used optimization algorithms, while maintaining a consistently lower number of objective function evaluations. Its stability across different functions and dimensions confirms its applicability to a broad range of optimization scenarios, making it a promising and efficient alternative within the field of evolutionary and metaheuristic optimization.

Table 2. Experimental results using different optimization methods. Numbers in cells represent sum function calls.

FUNCTION	BICCA	MLSHADESPA	SHADE_ILS	DE	GA	WOA	IPSO	PROPOSED
ATTRACTIVE SECTOR_25	5130	950	452	4439	2208	2641	2120	1697
ATTRACTIVE SECTOR_50	10097	994	558	18104	2230	5700	2167	1761
ATTRACTIVE SECTOR_100	20178	989	748	15246	2231	5785	2179	1832
ATTRACTIVE SECTOR_150	30259	1047	959	6646	2232	9248	2196	1867
BUCHE RASTRIGIN_25	5144(0.33)	9420(0.90)	2093(0.90)	1466(0.90)	12979(0.90)	15048(0.93)	12115(0.90)	5893(0.90)
BUCHE RASTRIGIN_50	10345(0.03)	18003(0.50)	3440(0.50)	1894(0.50)	20711(0.50)	58557(0.77)	30866(0.50)	12585(0.50)
BUCHE RASTRIGIN_100	20676(0.03)	30652(0.53)	5428(0.53)	2020(0.53)	29121(0.53)	43001(0.97)	39680(0.53)	16490(0.53)
BUCHE RASTRIGIN_150	30894(0.03)	47160(0.27)	7663(0.27)	2511(0.27)	37696(0.27)	54641	53060(0.27)	23466(0.27)
DISCUS_25	5125	1365	536	4255	2656	3006	2452	1992
DISCUS_50	10101	1425	642	10297	2663	6310	2498	2060
DISCUS_100	20189	1402	826	8284	2631	5835	2523	2104
DISCUS_150	30265	1487	1042	8548	2620	8227	2548	2144
DIFFERENTPOWERS_25	5144	13007	2644	4786	14495	14921	13313	6478
DIFFERENTPOWERS_50	10389	20029	3860	14391	20539	35828	19839	11183
DIFFERENTPOWERS_100	20644	27859	5450	7355	28413	52081	28379	16225
DIFFERENTPOWERS_150	30877	36894	7059	6266	33569	93074	36287	21495
ELLIPSOIDAL_25	5139(0.87)	4227	1117	4161	5955	7299	6375	3590
ELLIPSOIDAL_50	10247	9146	2178	16624	10892	19281	11641	6424
ELLIPSOIDAL_100	20492	18062	3966	12708	20202	38501	20736	11549
ELLIPSOIDAL_150	30708	26835	5993	21936	36236	63093	29414	16930
GALLAGHER21_25	5122(0.46)	1304(0.90)	503(0.90)	4180(0.90)	3346(0.90)	9210(0.90)	3605(0.90)	2261(0.90)
GALLAGHER21_50	10119(0.03)	1757(0.50)	701(0.50)	7938(0.50)	3192(0.50)	35580(0.50)	8866(0.50)	4503(0.50)
GALLAGHER21_100	20167	392(0.53)	637(0.53)	1323(0.53)	1593(0.53)	1950(0.53)	5363(0.53)	1756(0.53)
GALLAGHER21_150	30248	385(0.27)	825(0.27)	1313(0.27)	1582(0.27)	1738(0.27)	2050(0.27)	1662(0.27)
GALLAGHER101_25	5117(0.07)	1270	501(0.90)	3625(0.90)	3340(0.90)	7664(0.90)	3473(0.90)	2769(0.90)
GALLAGHER101_50	10114(0.03)	1396	634(0.50)	18470(0.50)	7134(0.50)	38817(0.50)	8796(0.50)	4890(0.50)
GALLAGHER101_100	20193(0.03)	1868	901(0.53)	14700(0.53)	5794(0.53)	39700(0.53)	9257(0.53)	5886(0.53)
GALLAGHER101_150	30269(0.03)	1922	1127(0.27)	24214(0.27)	7210(0.27)	36525(0.27)	14076(0.27)	8646(0.27)
GRIEWANK _25	5173(0.70)	7828	1811	4123(0.97)	9733	10166	9454	4084
GRIEWANK _50	10138	3434	1061	17524(0.93)	5410	18966	9827	5039
GRIEWANK _100	20208	2825	1124	14809	4982	19318	10369	6460
GRIEWANK _150	30290	3035	1391	6335(0.97)	5221	28823	10741	6542
GRIEWANK_ROSENBROCK_25	5180	14086	3132	3238	17038	10630	9698	4466
GRIEWANK_ROSENBROCK_50	10362	20021	4319	16379	23217	22912	11610	5325
GRIEWANK_ROSENBROCK_100	20462	23913	4925	11375	31195	24543	13409	6465
GRIEWANK_ROSENBROCK_150	30604	29813	6080	4446	37364	33948	15075	7272
ROSENBROCK_25	5163	12518	2793	3543	15493	13642	13642	5950
ROSENBROCK_50	10451	21195	4555	12085	24602	33038	22317	8963
ROSENBROCK_100	20785	35136	7151	6038	39496	48451	36400	15930
ROSENBROCK_150	31103	50850	10669	4203	53211	75425	50281	22135
RARSTIGIN_25	5139(0.36)	7826(0.90)	1767(0.90)	1574(0.90)	9581(0.90)	15530(0.90)	9826(0.90)	4577(0.90)
RARSTIGIN_50	10208(0.03)	10741(0.50)	2091(0.50)	1895(0.50)	12272(0.50)	41187(0.73)	17354(0.50)	7746(0.50)
RARSTIGIN_100	20358(0.03)	11464(0.53)	2338(0.53)	1869(0.53)	2134(0.53)	27383(0.90)	19347(0.53)	9147(0.53)
RARSTIGIN_150	30561(0.03)	14002(0.27)	2942(0.27)	2122(0.27)	13990(0.27)	32297(0.93)	27682(0.27)	11620(0.27)
SPHERE_25	5134	482	358	4131	1689	2206	1611	1481
SPHERE_50	10088	500	459	18098	1700	5111	1633	1509
SPHERE_100	20169	498	655	15241	1699	5107	1639	1524
SPHERE_150	30250	523	858	6639	1700	7347	1645	1535
STEP ELLIPSOIDAL_25	5114(0.70)	375(0.90)	313(0.90)	1857(0.93)	2069(0.90)	1812(0.97)	1846(0.90)	1625(0.90)
STEP ELLIPSOIDAL_50	10086(0.03)	375(0.50)	391(0.50)	6493(0.67)	2469(0.50)	2541(0.50)	3993(0.50)	2300(0.50)
STEP ELLIPSOIDAL_100	20167(0.03)	377(0.53)	541(0.53)	5658(0.53)	1681(0.53)	2405(0.53)	3946(0.53)	2465(0.53)
STEP ELLIPSOIDAL_150	30248(0.03)	383(0.27)	695(0.27)	5588(0.27)	1673(0.27)	2854(0.27)	5065(0.27)	3143(0.27)
SHARP RIDGE_25	5125	9281	2193	5153	11536	11398	11371	5104
SHARP RIDGE_50	10261	9843	2284	18677	11818	19405	12550	5226
SHARP RIDGE_100	20366	10190	2403	15159	11659	20507	13017	5995
SHARP RIDGE_150	30458	11205	2885	7476(0.97)	11866	25983	13776	6481
ZAKHAROV_25	5120	4383	1177	1735	5756	9556	3449	2185
ZAKHAROV_50	10118	18043	3584	2371	15522	23884	6469	3027
ZAKHAROV_100	20211	45770	8470	2216	38359	29581	16562	5572
ZAKHAROV_150	30293	46497	9315	2503	36399	32379	21273	6304
	992785(0.73)	708659(0.85)	157213(0.85)	478253(0.85)	786004(0.85)	1371596(0.90)	782751(0.85)	387335(0.85)

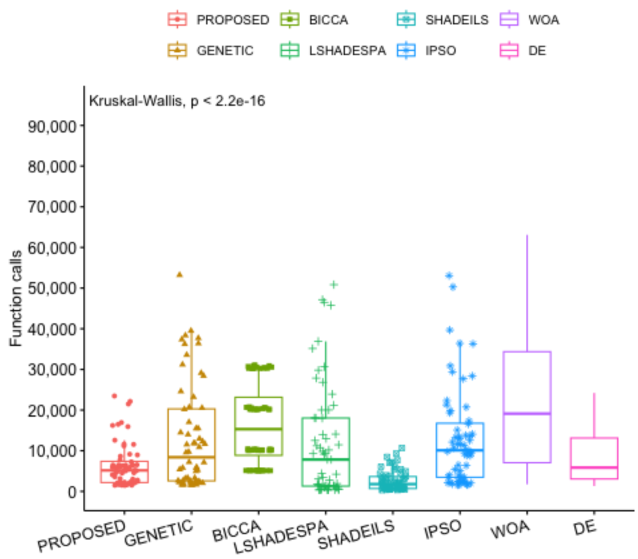

Figure 2. A statistical comparison of the proposed with other optimization methods.

Figure 2 illustrates the distribution of function evaluations for the PROPOSED optimizer compared with all baseline algorithms. A Kruskal–Wallis test confirms strong overall differences across methods ($p < 2.2e-16$), indicating that the optimizers do not come from the same underlying distribution. The PROPOSED method consistently exhibits substantially lower function-call counts, with both its median and dispersion markedly smaller than those of all competing algorithms. The results, combined with the consistently lower function-call requirements displayed in the figure, indicate that the superiority of the PROPOSED optimizer is highly unlikely to be attributable to random variation. Instead, it reflects a systematic and robust performance advantage over all baseline methods.

3.4. The effect of differential weight mechanism

Table 3 presents the impact of the three differential weight strategies NUMBER(T), RANDOM(T), and MIGRANT(T) on the performance of the algorithm across a broad set of benchmark functions, where (T) denotes tournament–based selection. The results clearly show that MIGRANT(T) is by far the most efficient method.. The MIGRANT(T) strategy consistently achieves the best outcomes, requiring the fewest objective function evaluations overall (387,335) compared with NUMBER(T) (527,444) and RANDOM(T) (543,201). This improvement is particularly noteworthy given that all three strategies achieve the same overall success rate (0.85), indicating that the performance advantage arises purely from efficiency rather than reliability differences. This trend is visible across multiple test functions. In the Attractive Sector family (25–150 dimensions), MIGRANT(T) demonstrates uniformly superior performance. For example, in Attractive Sector_25, it requires 1697 calls, compared with 1743 for NUMBER(T) and 1756 for RANDOM(T). As dimensionality increases, this advantage becomes even more pronounced. The performance gap becomes even more substantial in multimodal landscapes such as Buche–Rastrigin. In Buche Rastrigin_25, MIGRANT(T) needs 5893 function calls (success rate 0.90), significantly fewer than NUMBER(T) (12,243) and RANDOM(T) (12,035). The difference becomes overwhelming in the highest-dimensional case (Buche Rastrigin_150): MIGRANT(T) completes the optimization with 23,466 calls, while NUMBER(T) requires 40,240, and RANDOM(T) needs 39,263. These results underline MIGRANT(T)'s superior adaptability in sharply multimodal and high-variance landscapes. For example, Ellipsoidal_150 is solved in 16,930 calls by MIGRANT(T), compared with 19,311 for NUMBER(T) and 19,940 for RANDOM(T). This difference becomes particularly important for large-scale smooth problems, where maintaining efficiency is critical.

Overall, the evidence strongly indicates that MIGRANT(T) is the most effective differential weight mechanism among the tested variants. It consistently reduces the number of function evaluations across a wide variety of functions both unimodal and multimodal while preserving identical success rates. This combination of efficiency, robustness, and stability makes MIGRANT(T) a particularly advantageous choice for enhancing the performance of Differential Evolution in high-dimensional and challenging optimization scenarios.

Table 3. Experiments using different weight selection for the proposed method.

FUNCTION	MIGRANT (T)	NUMBER (T)	RANDOM (T)
ATTRACTIVE SECTOR_25	1697	1743	1756
ATTRACTIVE SECTOR_50	1761	1823	1828
ATTRACTIVE SECTOR_100	1832	1879	1880
ATTRACTIVE SECTOR_150	1867	1900	1920
BUCHE RASTRIGIN_25	5893(0.90)	12243(0.90)	12035(0.90)
BUCHE RASTRIGIN_50	12585(0.50)	19529(0.50)	20457(0.50)
BUCHE RASTRIGIN_100	16490(0.53)	30055(0.53)	31465(0.53)
BUCHE RASTRIGIN_150	23466(0.27)	40240(0.27)	39263(0.27)
DISCUS_25	1992	1857	1896
DISCUS_50	2060	1926	1971
DISCUS_100	2104	1978	1989
DISCUS_150	2144	2006	2040
DIFFERENTPOWERS_25	6478	11422	11629
DIFFERENTPOWERS_50	11183	15258	15179
DIFFERENTPOWERS_100	16225	21451	20659
DIFFERENTPOWERS_150	21495	24429	24670
ELLIPSOIDAL_25	3590	3751	3958
ELLIPSOIDAL_50	6424	6864	7184
ELLIPSOIDAL_100	11549	13756	13890
ELLIPSOIDAL_150	16930	19311	19940
GALLAGHER21_25	2261(0.90)	3815(0.90)	6364(0.90)
GALLAGHER21_50	4503(0.50)	5277(0.50)	9643(0.50)
GALLAGHER21_100	1756(0.53)	1523(0.53)	1521(0.53)
GALLAGHER21_150	1662(0.27)	1521(0.27)	1526(0.27)
GALLAGHER101_25	2769(0.90)	3472(0.90)	5657(0.90)
GALLAGHER101_50	4890(0.50)	6950(0.50)	7454(0.50)
GALLAGHER101_100	5886(0.53)	6846(0.53)	9505(0.53)
GALLAGHER101_150	8646(0.27)	7701(0.27)	12352(0.27)
GRIEWANK _25	4084	5276	5145
GRIEWANK _50	5039	5138	5729
GRIEWANK _100	6460	5726	6002
GRIEWANK _150	6542	5870	6164
GRIEWANK_ROSENBROCK_25	4466	7458	6939
GRIEWANK_ROSENBROCK_50	5325	9766	9255
GRIEWANK_ROSENBROCK_100	6465	11776	11001
GRIEWANK_ROSENBROCK_150	7272	13482	12543
ROSENBROCK_25	5950	7824	7955
ROSENBROCK_50	8963	13970	13057
ROSENBROCK_100	15930	23402	22348
ROSENBROCK_150	22135	32850	31562
RARSTIGIN_25	4577(0.90)	9691(0.90)	10242(0.90)
RARSTIGIN_50	7746(0.50)	13134(0.50)	12740(0.50)
RARSTIGIN_100	9147(0.53)	13128(0.53)	13184(0.53)
RARSTIGIN_150	11620(0.27)	15105(0.27)	15602(0.27)
SPHERE_25	1481	1507	1512
SPHERE_50	1509	1534	1539
SPHERE_100	1524	1555	1556
SPHERE_150	1535	1568	1567
STEP ELLIPSOIDAL_25	1625(0.90)	1642(0.90)	2090(0.90)
STEP ELLIPSOIDAL_50	2300(0.50)	2774(0.50)	4021(0.50)
STEP ELLIPSOIDAL_100	2465(0.53)	1598(0.53)	1571(0.53)
STEP ELLIPSOIDAL_150	3143(0.27)	1531(0.27)	1521(0.27)
SHARP RIDGE_25	5104	6215	6026
SHARP RIDGE_50	5226	6850	7123
SHARP RIDGE_100	5995	7782	7649
SHARP RIDGE_150	6481	8112	8237
ZAKHAROV_25	2185	2752	2639
ZAKHAROV_50	3027	4063	3864
ZAKHAROV_100	5572	6265	5634
ZAKHAROV_150	6304	7574	7553
	387335(0.85)	527444(0.85)	543201(0.85)

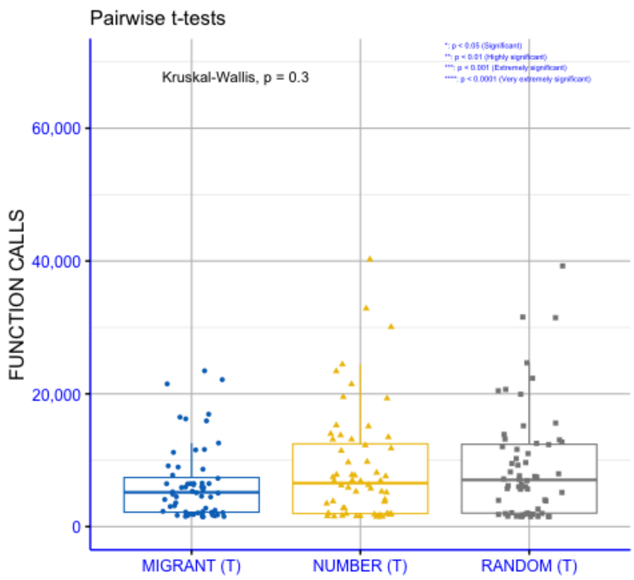

Figure 3. A statistical comparison of the proposed with different weight selection.

Figure 3 presents the pairwise statistical analysis of the three migration strategies MIGRANT (T), NUMBER (T), and RANDOM (T) based on their required function evaluations. According to the Kruskal–Wallis test (p = 0.3), no statistically significant overall difference is detected among the three strategies, indicating that their distributions of function calls are broadly comparable. Pairwise t-tests further support this conclusion: none of the comparisons reach conventional significance thresholds (ns: p > 0.05), demonstrating that the observed differences in median and variability across the strategies are not statistically meaningful. Although the MIGRANT (T) strategy tends to exhibit slightly lower function-call counts on average, these differences fall within the range of natural stochastic variation. Collectively, these findings suggest that the three strategies perform similarly with respect to computational cost.

3.5. The effect of selection mechanism

Table 4compares the four selection strategies RANDOM(R), RANDOM(T), MIGRANT(R), and MIGRANT(T), where (R) denotes purely random selection and (T) denotes tournament–based selection. The results clearly show that MIGRANT(T) is by far the most efficient method. It achieves the lowest total number of objective function evaluations (387,335), significantly outperforming MIGRANT(R) (962,599), RANDOM(T) (543,201), and RANDOM(R) (767,225). Since all methods achieve the same success rate (0.85), the performance differences are due solely to efficiency, demonstrating the importance of the selection mechanism. This advantage becomes evident across nearly all tested functions. For the Attractive Sector family (25–150 dimensions), MIGRANT(T) consistently requires the fewest evaluations. For example, in Attractive Sector_25, it needs only 1697 calls, compared to 2174 MIGRANT(R), 1756 for RANDOM(T) and as many as 2162 for RANDOM(R). The differences are even more striking for multimodal benchmarks such as Buche–Rastrigin. In the 25-dimensional case, MIGRANT(T) achieves 5893 calls (0.90 success), while MIGRANT(R) needs 15,894, RANDOM(T) needs 12,035, and RANDOM(R) 11,921. At the 150-dimensional level, the gap widens dramatically: MIGRANT(T) requires 23,466 calls, whereas MIGRANT(R) rises to 77,590, RANDOM(T) 39,263 and RANDOM(R) to 54,663. These results highlight the strong stabilizing effect that tournament selection has on the MIGRANT mechanism. The superiority of MIGRANT(T) is even more pronounced in the

Sharp Ridge functions. In Sharp Ridge_150, MIGRANT(T) completes the optimization with 6481 calls, while MIGRANT(R) requires 12,053, RANDOM(T) 8237, and RANDOM(R) 12,395. Finally, in the Zakharov functions, MIGRANT(T) again shows consistently superior performance. In Zakharov_150, MIGRANT(T) needs just 6304 calls, compared to 25,370 for MIGRANT(R), 7553 RANDOM(T) and 16,240 for RANDOM(R). Even in the easier 25-dimensional case, MIGRANT(T) requires 2185 calls, whereas RANDOM(R) requires over twice as many (4605).

Overall, these results highlight the strong interaction between the MIGRANT mechanism and tournament selection. Tournament selection dramatically enhances the performance of MIGRANT, reducing the computational cost by large margins across all functions while preserving identical success rates. As a result, MIGRANT(T) emerges as the most balanced, stable, and efficient strategy, making it highly suitable for optimization scenarios where minimizing objective function evaluations is essential.

Table 4. Effect of Random and Tournament Selection Strategies on Optimization Performance

FUNCTION	MIGRANT (T)	MIGRANT (R)	RANDOM (T)	RANDOM (R)
ATTRACTIVE SECTOR_25	1697	2174	1756	2162
ATTRACTIVE SECTOR_50	1761	2212	1828	2162
ATTRACTIVE SECTOR_100	1832	2177	1880	2192
ATTRACTIVE SECTOR_150	1867	2206	1920	2174
BUCHE RASTRIGIN_25	5893(0.90)	15894(0.90)	12035(0.90)	11921(0.90)
BUCHE RASTRIGIN_50	12585(0.50)	50438(0.50)	20457(0.50)	20542(0.50)
BUCHE RASTRIGIN_100	16490(0.53)	59214(0.53)	31465(0.53)	34570(0.53)
BUCHE RASTRIGIN_150	23466(0.27)	77590(0.27)	39263(0.27)	54663(0.27)
DISCUS_25	1992	2588	1896	2542
DISCUS_50	2060	2601	1971	2552
DISCUS_100	2104	2553	1989	2617
DISCUS_150	2144	2608	2040	2591
DIFFERENTPOWERS_25	6478	13918	11629	14477
DIFFERENTPOWERS_50	11183	20100	15179	20064
DIFFERENTPOWERS_100	16225	27396	20659	29408
DIFFERENTPOWERS_150	21495	35710	24670	35070
ELLIPSOIDAL_25	3590	6424	3958	5932
ELLIPSOIDAL_50	6424	11704	7184	10585
ELLIPSOIDAL_100	11549	20736	13890	20887
ELLIPSOIDAL_150	16930	29835	19940	28265
GALLAGHER21_25	2261(0.90)	5412(0.90)	6364(0.90)	2891(0.90)
GALLAGHER21_50	4503(0.50)	11988(0.50)	9643(0.50)	3311(0.50)
GALLAGHER21_100	1756(0.53)	1565(0.53)	1521(0.53)	1524(0.53)
GALLAGHER21_150	1662(0.27)	1490(0.27)	1526(0.27)	1520(0.27)
GALLAGHER101_25	2769(0.90)	5180(0.90)	5657(0.90)	3016(0.90)
GALLAGHER101_50	4890(0.50)	21179(0.50)	7454(0.50)	4878(0.50)
GALLAGHER101_100	5886(0.53)	26739(0.53)	9505(0.53)	4507(0.53)
GALLAGHER101_150	8646(0.27)	46866(0.27)	12352(0.27)	3400(0.27)
GRIEWANK _25	4084	8148	5145	9902
GRIEWANK _50	5039	7894	5729	5203
GRIEWANK _100	6460	9083	6002	4145
GRIEWANK _150	6542	9154	6164	4075
GRIEWANK_ROSENBROCK_25	4466	11510	6939	17429
GRIEWANK_ROSENBROCK_50	5325	14658	9255	24666
GRIEWANK_ROSENBROCK_100	6465	15890	11001	34019
GRIEWANK_ROSENBROCK_150	7272	17910	12543	39208
ROSENBROCK_25	5950	13718	7955	15591
ROSENBROCK_50	8963	21827	13057	23980
ROSENBROCK_100	15930	34948	22348	40245
ROSENBROCK_150	22135	49061	31562	53073
RARSTIGIN_25	4577(0.90)	11276(0.90)	10242(0.90)	9910(0.90)
RARSTIGIN_50	7746(0.50)	26967(0.50)	12740(0.50)	14234(0.50)
RARSTIGIN_100	9147(0.53)	27639(0.53)	13184(0.53)	16666(0.53)
RARSTIGIN_150	11620(0.27)	34865(0.27)	15602(0.27)	19135(0.27)
SPHERE_25	1481	1620	1512	1627
SPHERE_50	1509	1641	1539	1634
SPHERE_100	1524	1635	1556	1644
SPHERE_150	1535	1644	1567	1639
STEP ELLIPSOIDAL_25	1625(0.90)	2073(0.90)	2090(0.90)	1750(0.90)
STEP ELLIPSOIDAL_50	2300(0.50)	5937(0.50)	4021(0.50)	1664(0.50)
STEP ELLIPSOIDAL_100	2465(0.53)	6546(0.53)	1571(0.53)	1523(0.53)
STEP ELLIPSOIDAL_150	3143(0.27)	11487(0.27)	1521(0.27)	1520(0.27)
SHARP RIDGE_25	5104	10153	6026	11776
SHARP RIDGE_50	5226	11108	7123	12123
SHARP RIDGE_100	5995	11592	7649	12704
SHARP RIDGE_150	6481	12053	8237	12395
ZAKHAROV_25	2185	3941	2639	4605
ZAKHAROV_50	3027	8972	3864	7963
ZAKHAROV_100	5572	23782	5634	14514
ZAKHAROV_150	6304	25370	7553	16240
	387335(0.85)	962599(0.85)	543201(0.85)	767225(0.85)

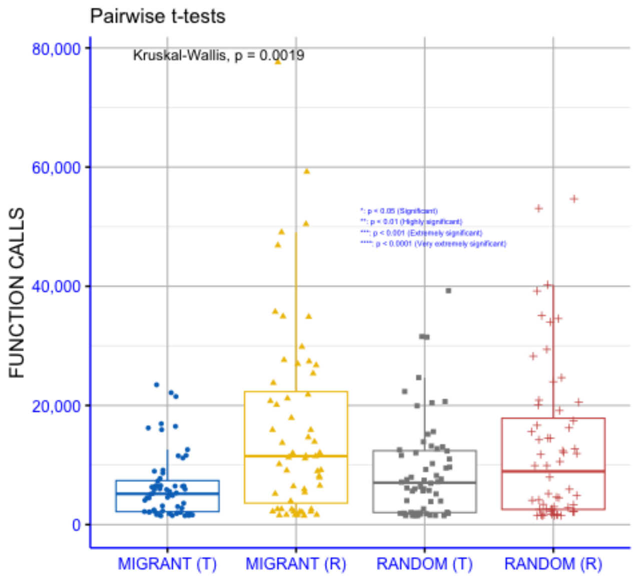

Figure 4. Statistical Comparison of Random and Tournament Selection Strategies on Optimization Performance.

Figure 4 presents the pairwise statistical analysis between the four strategies MIGRANT(T), MIGRANT(R), RANDOM(T), and RANDOM(R) based on their function-call distributions. A Kruskal–Wallis test reports a statistically significant overall difference among the groups (p = 0.0019), indicating that at least one strategy differs meaningfully from the others. To further investigate these differences, pairwise t-tests were conducted, and the resulting p-values were annotated using the standard significance notation (ns: p > 0.05, : p < 0.05, *: p < 0.01, ***: p < 0.001, ****: p < 0.0001). The results show that MIGRANT(T) performs significantly better than both MIGRANT(R) and RANDOM(R), achieving **level differences against the highest-cost configuration. Additionally, MIGRANT(R) and RANDOM(T) exhibit intermediate performance, with several comparisons reaching or *** significance levels, indicating meaningful but less pronounced differences. Comparisons classified as "ns" suggest that some pairs have statistically indistinguishable behavior. Overall, the distribution of function calls shows that the MIGRANT-based strategies, particularly MIGRANT(T), tend to require fewer evaluations, demonstrating stronger computational efficiency relative to the RANDOM variants.

3.6. The effect of sampling method

In Table 5 tournament selection is used to choose the samples that participate in the core operator of Differential Evolution. Four strategies for computing the differential weight are evaluated: random weight with uniform sampling (Random(U)), random weight with k-means sampling (Random(K)), MIGRANT weight with uniform sampling (Migrant(U)), and MIGRANT weight with k-means sampling (Migrant(K)). The k-means method, originally proposed by MacQueen[61] and used extensively in later work [65, 66], is employed not only to determine cluster centers but also as a structured sampling mechanism. Across all test functions, MIGRANT(K) consistently achieves the lowest total number of function calls (387,335) with a success rate of 0.85, outperforming all other sampling strategies. This advantage becomes clear when examining individual benchmarks. For the Attractive Sector family (dimensions 25–150), MIGRANT(K) systematically requires fewer evaluations than MIGRANT(U) and both Random methods. For example, in Attractive Sector_25, MIGRANT(K) uses only 1697 evaluations compared to 1738for

MIGRANT(U), while Random(K) and Random(U) require 1756 and 1792 respectively. This pattern holds across all dimensionalities, showing the benefit of structured sampling in unimodal landscapes. The effect becomes far more pronounced in multimodal functions such as Buche–Rastrigin. In Buche Rastrigin_25, MIGRANT(K) needs 5893 function calls with a success rate of 0.90, in contrast to MIGRANT(U)'s 12,818 calls (0.03). Random(K) performs similarly to MIGRANT(K) in success rate but requires more evaluations (12,035), while Random(U) is by far the least efficient (28,865 calls). The difference becomes dramatic in higher dimensions: in Buche Rastrigin_150, MIGRANT(K) performs the task in 23,466 calls (0.27), whereas Random(U) escalates to 90,211, showing the instability of uniform sampling in complex landscapes. Although differences here are smaller due to the problem's structure, MIGRANT(K) preserves its advantage in stability. The superiority of MIGRANT(K) is again evident in the Step Ellipsoidal group. In Step Ellipsoidal_25, MIGRANT(K) requires only 5104 calls, remarkably lower than Random(U) (6699), Random (K) (6026) and still better than MIGRANT(U) (5014, but with lower success). Finally, for the Zakharov functions, MIGRANT(K) again shows the best balance between evaluation cost and success rate. In Zakharov_25, MIGRANT(K) requires only 2185 evaluations, beating Migrant(U) (2283), Random(K) (2639) and Random(U) (2797).

Overall, integrating k-means sampling into the MIGRANT strategy leads to substantial improvements in both efficiency and reliability. MIGRANT(K) not only requires the fewest total function evaluations but also maintains high success rates across diverse problem categories, making it the most effective approach for the benchmark set. In contrast, Random(U) repeatedly demonstrates the lowest efficiency, highlighting the advantage of structured sampling over uniform dispersion in high-dimensional optimization.

Table 5. Experiments on the performance of differential evolution using sampling methods

FUNCTION	MIGRANT (K)	MIGRANT (U)	RANDOM (K)	RANDOM (U)
ATTRACTIVE SECTOR_25	1697	1738	1756	1792
ATTRACTIVE SECTOR_50	1761	1792	1828	1832
ATTRACTIVE SECTOR_100	1832	1866	1880	1891
ATTRACTIVE SECTOR_150	1867	1890	1920	1920
BUCHE RASTRIGIN_25	5893(0.90)	12818(0.03)	12035(0.90)	28865(0.03)
BUCHE RASTRIGIN_50	12585(0.50)	23622(0.03)	20457(0.50)	34379(0.03)
BUCHE RASTRIGIN_100	16490(0.53)	41526(0.03)	31465(0.53)	70319(0.03)
BUCHE RASTRIGIN_150	23466(0.27)	55612(0.03)	39263(0.27)	90211(0.03)
DIFFERENT POWERS_25	1992	2016	1896	1936
DIFFERENT POWERS_50	2060	2077	1971	1989
DIFFERENT POWERS_100	2104	2114	1989	2026
DIFFERENT POWERS_150	2144	2150	2040	2058
DISCUS_25	6478	7368	11629	11484
DISCUS_50	11183	11666	15179	15789
DISCUS_100	16225	17566	20659	21459
DISCUS_150	21495	22526	24670	24485
ELLIPSOIDAL_25	3590	3640	3958	3873
ELLIPSOIDAL_50	6424	6399	7184	7022
ELLIPSOIDAL_100	11549	12161	13890	13610
ELLIPSOIDAL_150	16930	17905	19940	19576
GALLAGHER21_25	2261(0.90)	6920(0.03)	6364(0.90)	34112(0.03)
GALLAGHER21_50	4503(0.50)	7904(0.03)	9643(0.50)	17404(0.03)
GALLAGHER21_100	1756(0.53)	1463	1521(0.53)	1524(0.53)
GALLAGHER21_150	1662(0.27)	1463	1526(0.27)	1522(0.27)
GALLAGHER101_25	2769(0.90)	6395(0.03)	5657(0.90)	27324(0.03)
GALLAGHER101_50	4890(0.50)	8204(0.03)	7454(0.50)	17075(0.03)
GALLAGHER101_100	5886(0.53)	10816(0.03)	9505(0.53)	18232(0.03)
GALLAGHER101_150	8646(0.27)	12129(0.03)	12352(0.27)	17231(0.03)
GRIEWANK ROSENBROCK_25	4084	4353	5145	5434
GRIEWANK ROSENBROCK_50	5039	5290	5729	5631
GRIEWANK ROSENBROCK_100	6460	6211	6002	5916
GRIEWANK ROSENBROCK_150	6542	6895	6164	6113
GRIEWANK_25	4466	4818	6939	7697
GRIEWANK_50	5325	7163	9255	11056
GRIEWANK_100	6465	9992	11001	15311
GRIEWANK_150	7272	12350	12543	19125
RARSTIGIN_25	5950	5909(0.03)	7955	8447(0.03)
RARSTIGIN_50	8963	10112(0.03)	13057	13669(0.03)
RARSTIGIN_100	15930	16541(0.03)	22348	23760(0.03)
RARSTIGIN_150	22135	23181(0.03)	31562	33005(0.03)
ROSENBROCK_25	4577(0.90)	9432	10242(0.90)	18663
ROSENBROCK_50	7746(0.50)	11863	12740(0.50)	27806
ROSENBROCK_100	9147(0.53)	15307	13184(0.53)	28064
ROSENBROCK_150	11620(0.27)	18904	15602(0.27)	43292
SHARP RIDGE_25	1481	1498	1512	1528
SHARP RIDGE_50	1509	1516	1539	1548
SHARP RIDGE_100	1524	1531	1556	1559
SHARP RIDGE_150	1535	1548	1567	1565
SPHERE_25	1625(0.90)	2733	2090(0.90)	7103
SPHERE_50	2300(0.50)	3173	4021(0.50)	6384
SPHERE_100	2465(0.53)	3654	1571(0.53)	5873
SPHERE_150	3143(0.27)	4073	1521(0.27)	5149
STEP ELLIPSOIDAL_25	5104	5014(0.03)	6026	6699(0.03)
STEP ELLIPSOIDAL_50	5226	5581(0.03)	7123	7205(0.03)
STEP ELLIPSOIDAL_100	5995	6091(0.03)	7649	7893(0.03)
STEP ELLIPSOIDAL_150	6481	5996(0.03)	8237	8037(0.03)
ZAKHAROV_25	2185	2283	2639	2797
ZAKHAROV_50	3027	2901	3864	3743
ZAKHAROV_100	5572	4122	5634	5936
ZAKHAROV_150	6304	5282	7553	7460
	387335(0.85)	529063(0.71)	543201(0.85)	844408(0.69)

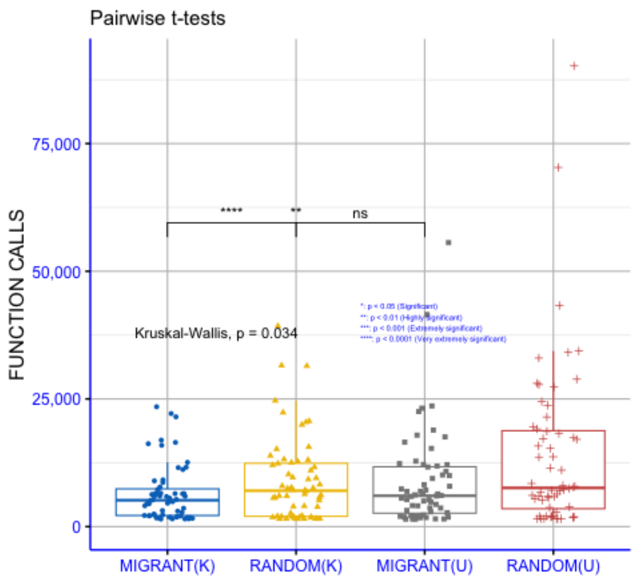

Figure 5. Statistical Comparison of Different Sampling Method Combinations in Differential Evolution Performance.

Figure 5 presents the pairwise t-tests statistical comparison of the four strategies (MIGRANT(K), RANDOM(K), MIGRANT(U), and RANDOM(U)) based on their function-call distributions. A Kruskal–Wallis test indicates a statistically significant overall difference among the groups (p = 0.034), suggesting that at least one strategy exhibits a distinct performance profile. Pairwise t-tests, corrected using the conventional significance notation (ns: $p > 0.05$, : $p < 0.05$, *: $p < 0.01$, *: $p < 0.001$, : $p < 0.0001$), reveal that MIGRANT(K) differs very significantly from RANDOM(K) and significantly from MIGRANT(U). In contrast, the difference between MIGRANT(U) and RANDOM(U) is not statistically significant (ns). Taken together, these results show that strategies based on migration with K-type selection (MIGRANT(K)) demonstrate consistently lower function-call requirements, while the U-type variants display comparable behavior to the RANDOM(U) baseline. Overall, the statistical evidence suggests that the K-based migration mechanism yields a measurable performance advantage relative to the other strategies.

3.7. The effect of local search rate

From Table 6 we observe the influence of periodic local optimization on the performance of the MIGRANT method, considering four different local search rates: 0.005, 0.01, 0.03, and 0.05. Among all settings, the 0.005 rate achieves the lowest total number of function calls (148,027) while maintaining a high success rate of 0.85, thus providing the best balance between computational efficiency and optimization reliability. This advantage is consistently reflected across the benchmark functions. In the Attractive Sector functions (25–150 dimensions), the 0.005 rate clearly outperforms the higher-rate configurations. For instance, in Attractive Sector_25, it requires only 1441 calls, compared to 1603 for the 0.03 rate and 1697 for the 0.05 rate. The improvement persists as the dimensionality increases: Attractive Sector_150 is solved with 1536 calls at the 0.005 rate, while the 0.05 rate requires 1867 calls. The improvement becomes dramatically more pronounced in the Buche–Rastrigin family, where the complexity and multimodality amplify the benefit of lower local search frequency. For Buche Rastrigin_25, the 0.005 method requires 2035 function calls (0.90 success), whereas the 0.05 rate jumps to 5893 calls nearly triple. In the high-dimensional case Buche Rastrigin_150, the difference is even more striking: 5900 calls at the 0.005 rate versus 23,466 for the 0.05 rate. A similar trend can be seen in the Discus, Sharp Ridge, and Step Ellipsoidal functions. In Step Ellipsoidal_50, the 0.005 rate achieves 2136 calls (0.50), far below the 0.05 rate (2300). For Step Ellipsoidal_150, the 0.005 variant uses 2914 calls (0.27), while the 0.05 rate needs 3143 calls. Even in unimodal functions like Discus, the 0.005 method consistently leads to lower evaluation costs e.g., Discus_25 requires 1525 calls vs. 1992 for the 0.05 rate. The Sharp Ridge functions highlight this behavior even more strongly. For Sharp Ridge_25, the 0.005 rate requires only 1934 calls, in contrast to 5104 calls for the 0.05 rate more than a 2.5× increase. Similar improvements appear in Sharp Ridge_150, where the function calls rise from 2350 at 0.005 to 6481 at 0.05.

In summary, using a lower local search rate specifically the 0.005 setting results in the most efficient optimization behavior across all tested functions. This variant provides the lowest objective function calls without compromising success rate, making it the optimal choice when both efficiency and reliability are essential in high-dimensional optimization tasks.

Table 6. Experiments on the Effect of Local Search Rate on Optimization Performance in Differential Evolution.

FUNCTION	MIGRANT(0.005)	MIGRANT(0.01)	MIGRANT(0.03)	MIGRANT(0.05)
ATTRACTIVE SECTOR_25	1441	1472	1603	1697
ATTRACTIVE SECTOR_50	1582	1522	1674	1761
ATTRACTIVE SECTOR_100	1516	1552	1699	1832
ATTRACTIVE SECTOR_150	1536	1560	1726	1867
BUCHE RASTRIGIN_25	2035(0.90)	2502(0.90)	4323(0.90)	5893(0.90)
BUCHE RASTRIGIN_50	3468(0.50)	4319(0.50)	8496(0.50)	12585(0.50)
BUCHE RASTRIGIN_100	4179(0.53)	5700(0.53)	10756(0.53)	16490(0.53)
BUCHE RASTRIGIN_150	5900(0.27)	7794(0.27)	14818(0.27)	23466(0.27)
DISCUS_25	1525	1616	1841	1992
DISCUS_50	1615	1658	1919	2060
DISCUS_100	1578	1655	1979	2104
DISCUS_150	1590	1663	1987	2144
DIFFERENTPOWERS_25	2296	2855	4661	6478
DIFFERENTPOWERS_50	3011	3807	7580	11183
DIFFERENTPOWERS_100	3827	5693	11214	16225
DIFFERENTPOWERS_150	4736	7158	15238	21495
ELLIPSOIDAL_25	1765	2011	2940	3590
ELLIPSOIDAL_50	2235	2844	4854	6424
ELLIPSOIDAL_100	3234	4557	9215	11549
ELLIPSOIDAL_150	4581	6620	12510	16930
GALLAGHER21_25	1751(0.90)	1804(0.90)	2049(0.90)	2261(0.90)
GALLAGHER21_50	2842(0.50)	3012(0.50)	3765(0.50)	4503(0.50)
GALLAGHER21_100	1432(0.53)	1470(0.53)	1609(0.53)	1756(0.53)
GALLAGHER21_150	1434(0.27)	1455(0.27)	1554(0.27)	1662(0.27)
GALLAGHER101_25	1778(0.90)	1896(0.90)	2359(0.90)	2769(0.90)
GALLAGHER101_50	3287(0.50)	3470(0.50)	4186(0.50)	4890(0.50)
GALLAGHER101_100	3550(0.53)	3804(0.53)	4851(0.53)	5886(0.53)
GALLAGHER101_150	4726(0.27)	5208(0.27)	6959(0.27)	8646(0.27)
GRIEWANK _25	1858	2102	3137	4084
GRIEWANK _50	2154	2407	3859	5039
GRIEWANK _100	2135	2688	4542	6460
GRIEWANK _150	2298	2937	4919	6542
GRIEWANK_ROSENBROCK_25	1840	2116	3292	4466
GRIEWANK_ROSENBROCK_50	2091	2661	4199	5325
GRIEWANK_ROSENBROCK_100	2343	2969	4868	6465
GRIEWANK_ROSENBROCK_150	2512	3295	5496	7272
ROSENBROCK_25	1964	2450	4091	5950
ROSENBROCK_50	2675	3619	6578	8963
ROSENBROCK_100	3616	5278	10570	15930
ROSENBROCK_150	5326	7819	15572	22135
RARSTIGIN_25	1831(0.90)	2135(0.90)	3346(0.90)	4577(0.90)
RARSTIGIN_50	2858(0.50)	3334(0.50)	5664(0.50)	7746(0.50)
RARSTIGIN_100	2941(0.53)	3697(0.53)	6336(0.53)	9147(0.53)
RARSTIGIN_150	3997(0.27)	4826(0.27)	8275(0.27)	11620(0.27)
SPHERE_25	1402	1411	1455	1481
SPHERE_50	1537	1444	1475	1509
SPHERE_100	1463	1454	1489	1524
SPHERE_150	1481	1469	1494	1535
STEP ELLIPSOIDAL_25	1513(0.90)	1526(0.90)	1576(0.90)	1625(0.90)
STEP ELLIPSOIDAL_50	2136(0.50)	2155(0.50)	2229(0.50)	2300(0.50)
STEP ELLIPSOIDAL_100	2286(0.53)	2308(0.53)	2389(0.53)	2465(0.53)
STEP ELLIPSOIDAL_150	2914(0.27)	2938(0.27)	3040(0.27)	3143(0.27)
SHARP RIDGE_25	1934	2269	3453	5104
SHARP RIDGE_50	2130	2680	4042	5226
SHARP RIDGE_100	2190	2718	4489	5995
SHARP RIDGE_150	2350	3107	5131	6481
ZAKHAROV_25	1570	1635	1912	2185
ZAKHAROV_50	1714	1884	2505	3027
ZAKHAROV_100	2428	2677	4597	5572
ZAKHAROV_150	2090	2314	4721	6304
	148027(0.85)	178999(0.85)	289106(0.85)	387335(0.85)

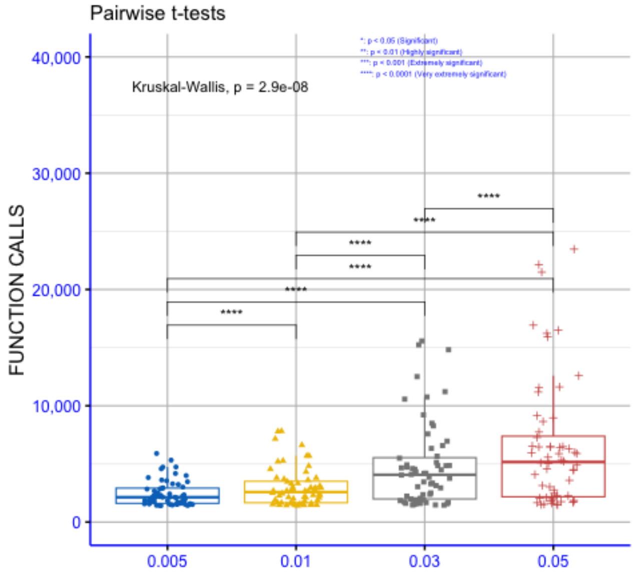

Figure 6. Statistical comparison for the proposed method and different values of parameter p_l.

Figure 6 presents the pairwise statistical comparisons among the four parameter configurations (0.001, 0.01, 0.03, 0.05) using Pairwise t-tests. The global Kruskal–Wallis test indicates a statistically significant difference across the groups (p = 2.9e-08), suggesting that the parameter choice has a measurable impact on the number of function evaluations.

All pairwise comparisons were evaluated using independent t-tests, and the resulting p-values were interpreted using the conventional significance notation (ns: $p > 0.05$, : $p < 0.05$, *: $p < 0.01$, ***: $p < 0.001$, **: $p < 0.0001$). As illustrated in the figure, most pairwise differences reach * or **** levels of significance, highlighting strong and highly consistent differences between the examined parameter settings. Only a limited number of comparisons fall into the non-significant range, indicating that in the majority of cases each parameter configuration leads to distinguishable performance in terms of function calls. These results confirm that the tested parameter values influence the optimizer's efficiency in a systematic and statistically robust manner. The differences detected are unlikely to be attributed to random variation and instead reflect meaningful performance changes caused by the selected parameter settings.

3.8. Practical problems

To further examine the practical efficiency and scalability of the proposed optimization algorithm, two real-world engineering design problems were investigated: the GasCycle[77] and the Tandem Queueing System[78]. These problems were selected because they differ significantly in mathematical formulation and computational complexity, providing a comprehensive framework for evaluating the algorithm's performance under diverse and realistic conditions.

Each problem was tested across multiple dimensional configurations, ranging from 25 to 500 variables, in order to assess how the algorithm behaves as the search space becomes more complex. For every configuration, the execution time in seconds was recorded as the main performance indicator. This experimental setup enables a direct comparison of how computational efficiency changes with increasing dimensionality.

- **GasCycle Thermal Cycle**

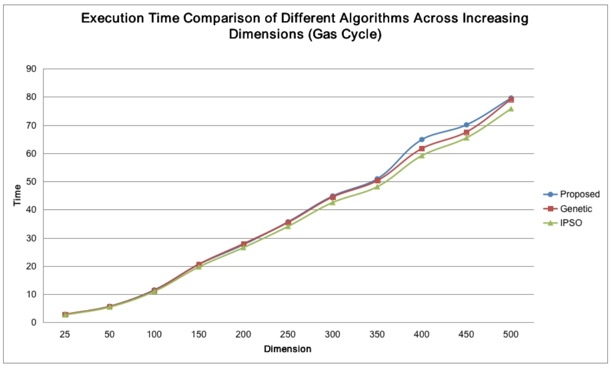

Figure 7. Execution Time Comparison of Different Algorithms Across Increasing Dimensions(GasCycle)

Vars: $\boldsymbol{x} = [T_1, T_3, P_1, P_3]^\top$. $\quad r = P_3/P_1,\ \gamma = 1.4.$

$$\eta(\boldsymbol{x}) = 1 - r^{-(\gamma-1)/\gamma}\,\frac{T_1}{T_3}, \qquad \min_{\boldsymbol{x}} f(\boldsymbol{x}) = -\eta(\boldsymbol{x}).$$

Bounds: $300 \le T_1 \le 1500,\ 1200 \le T_3 \le 2000,\ 1 \le P_1, P_3 \le 20.$
Penalty: infeasible $\Rightarrow f = 10^{20}$.
The experimental evaluation of the three algorithms demonstrates a predictable increase in execution time as the dimensionality of the problem grows, which aligns with the expected computational complexity of population-based optimization methods. In lower and medium dimensions, the algorithms exhibit highly comparable behavior, with only minor fluctuations that reflect inherent differences in their internal search mechanisms. As the dimensionality increases, more noticeable distinctions begin to emerge. In certain higher-dimensional cases, the proposed algorithm requires a longer execution time compared to the other two approaches. This outcome can be attributed to the structural characteristics and computational demands of its individual components, without implying any deficiency in its design. On the contrary, the overall performance profile of the proposed method remains consistent and closely aligned with that of the Genetic and IPSO algorithms, indicating that it scales reliably even as the dimensionality becomes substantially larger. Overall, the results show that all methods behave robustly with respect to execution time, with variations that are reasonable and expected given their algorithmic properties. The observed differences do not affect the general conclusion that the proposed algorithm falls well within the performance range of established techniques and maintains dependable behavior across the full spectrum of tested dimensions.

- **Tandem Space Trajectory** (MGA-1DSM, EVEEJ + 2×Saturn)
Vars (**D=18**): $\boldsymbol{x} = [t_0, T_1, T_2, T_3, T_4, T_{5A}, T_{5B}, s_1, s_2, s_3, s_4, s_{5A}, s_{5B}, r_p, k_{A1}, k_{A2}, k_{B1}, k_{B2}]^\top$.

$$\begin{aligned}
&7000 \le t_0 \le 10000,\\
&30 \le T_1 \le 500,\ 30 \le T_2 \le 600,\ 30 \le T_3 \le 1200,\\
&30 \le T_4 \le 1600,\ 30 \le T_{5A}, T_{5B} \le 2000,\\
&0 \le s_{1..4}, s_{5A}, s_{5B}, r_p, k_{A1}, k_{A2}, k_{B1}, k_{B2} \le 1.
\end{aligned}$$

Objective:

$$\min_{\boldsymbol{x}}\ \Delta V_{\text{tot}} = \Delta V_{\text{launch}}(T_1) + \Delta V_{\text{legs}}(T_1{:}T_4) + \Delta V_A + \Delta V_B + \Delta V_{\text{DSM}}(\boldsymbol{s}, r_p) - G_{\text{GA}} - G_J + P_{\text{hard}} +$$

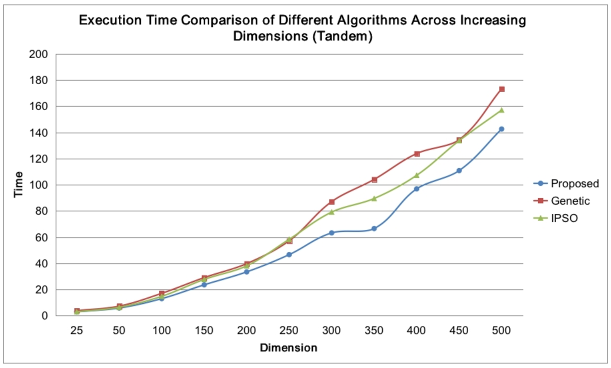

Figure 8. Execution Time Comparison of Different Algorithms Across Increasing Dimensions(Tandem)

$$P_{\text{soft}} = \beta \max\Big\{0,\ (T_1 + \cdots + T_4 + \tfrac{1}{2}(T_{5A} + T_{5B})) - 3500\Big\}.$$

Notes: ΔV_{launch} decreases (log-like) in T_1 (≥ 6 km/s floor), leg/branch costs decrease with TOF.

The execution time analysis for the Tandem problem demonstrates that the Proposed algorithm consistently achieves lower computational time across all tested dimensions when compared with the Genetic and IPSO methods. As the dimensionality increases, all algorithms exhibit the expected upward trend in execution time however, the Proposed approach maintains a steady advantage throughout the entire range. This consistent performance suggests that the internal structure and optimization mechanisms of the Proposed algorithm enable more efficient scaling, resulting in reduced computational cost even in higher-dimensional settings. Overall, the comparative results highlight the robustness and efficiency of the Proposed method, confirming its suitability for problems with increasing complexity.

4. Discussion

The results across all experiments paint a very clear picture: the choices we make in sampling, weighting, selection, and local search frequency strongly influence how the algorithm behaves. What becomes immediately noticeable is that when these components are designed in a structured and thoughtful way, the algorithm becomes not only faster but also far more stable.

1. Sampling Techniques

One of the most striking findings comes from the sampling study. Using k-means to guide the sampling process consistently leads to better performance than uniform sampling. The MIGRANT(K) configuration almost always requires fewer function calls and shows much tighter distributions. In contrast, uniform sampling often spreads the search in less helpful directions, leading to wasteful evaluations and higher variance. The pairwise t-tests confirm this: many comparisons involving MIGRANT(K) are not just significant, but highly significant.

2. Differential weight selection

The differential weight mechanism also plays a major role. The MIGRANT(T) method, which adapts the weight using information gathered during the search, outperforms both the NUMBER and RANDOM strategies. The fact that all three mechanisms achieve the same success rate makes this difference even more meaningful: MIGRANT(T) simply gets the job done with fewer evaluations. This behavior shows that giving the algorithm a way to "learn" more effectively from its population leads to smarter, more economical steps through the search space.

3. Selection Mechanisms

The impact of the selection mechanism is equally unmistakable. Tournament selection clearly strengthens the performance of the MIGRANT strategy. MIGRANT(T) is the best-performing approach across almost all functions, whereas MIGRANT(R) is often among the weakest. This suggests that random selection, despite its simplicity, can easily disrupt the search by favoring poor candidates. Tournament selection adds a light but meaningful pressure toward good solutions, which helps guide the algorithm in a more reliable direction.

4. Local Search Rate

The local search experiments reveal something quite intuitive: applying local search too frequently can actually hurt performance. The lowest rate, 0.005, consistently delivers the best results. Higher rates not only increase the cost dramatically but also introduce more noise into the optimization process. The Kruskal–Wallis p-values back this up, showing clear statistical differences between the rates. Essentially, a small and controlled dose of local search works best anything more becomes an unnecessary overhead.

5. Comparison with other Algorithms

When comparing the proposed approach to well-known methods such as GA, BICCA, LSHADE-SPA, SHADE-ILS, IPSO, WOA, and classical DE, the difference is visually and statistically unmistakable. The boxplots show the proposed method forming a compact, low-cost cluster, while competing algorithms display larger medians and far more variability. The extremely small p-values (e.g., $< 2.2e{-16}$) confirm that these differences are not random. The proposed method is simply more efficient and more reliable across the board. Putting everything together, a common theme emerges: the algorithm performs best when it combines structured exploration (via k-means sampling and tournament selection) with adaptive exploitation (through MIGRANT weighting and a low-rate local search). This blend gives the method a kind of "balance" that many classical and modern optimizers struggle to achieve. It doesn't rush into local optima, but it also doesn't waste evaluations wandering aimlessly.

Overall, the findings show that the proposed approach is not just marginally better it is consistently stronger, more efficient, and more stable across a wide range of benchmark functions. The improvements come from thoughtful design choices rather than brute-force complexity, making the method both elegant and practical. In many ways, the results highlight a simple but powerful idea: intelligent structure beats randomness. When the algorithm is given meaningful guidance through sampling, weighting, and selection it becomes a far more capable optimizer.

5. Conclusions

This work explored large-scale optimization through a systematically enhanced version of the Differential Evolution algorithm. The improvements introduced in this study were designed to address two persistent challenges in high-dimensional optimization: efficiency and stability. Throughout the experimental analysis, several key components proved crucial to achieving these goals. A central contribution is the MIGRANT differential weight mechanism, which consistently outperformed both the classic NUMBER and RANDOM schemes. Across a wide variety of benchmark functions, MIGRANT(T) required significantly fewer objective function evaluations while maintaining identical success rates. This demonstrates that an adaptive weight strategy can guide the search more intelligently, reducing unnecessary evaluations and offering clear performance advantages in complex landscapes. Equally important was the impact of the sampling strategy. The results showed that k-means sampling (K) provides a strong structural advantage compared to uniform sampling. Configurations using MIGRANT(K) repeatedly achieved the lowest evaluation counts and exhibited far smaller variance. Pairwise statistical tests confirmed these differences, with several comparisons reaching high or very high levels of significance. This indicates that exploiting cluster information during sampling can greatly improve the quality and diversity of candidate solutions. The study also highlighted the role of the selection mechanism. Tournament selection consistently strengthened the algorithm's performance, enabling MIGRANT(T) to outperform all Random-based variants. This confirms that introducing even a light degree of selective pressure yields more reliable search dynamics, while fully random selection tends to increase noise and computational cost. Another important outcome relates to the local search rate. Although local search can refine promising candidates, the experiments showed that applying it too frequently becomes counterproductive. The lowest tested rate (0.005) offered the best trade-off, achieving lower computational cost and greater stability. In contrast, higher rates (0.03 and 0.05) significantly increased function evaluations without improving success rates. This emphasizes the need for careful calibration of exploitation mechanisms in high-dimensional settings. Finally, when compared to widely used algorithms such as Genetic Algorithms, BICCA, LSHADE-SPA, SHADE-ILS, IPSO, WOA, and DE, the proposed method consistently delivered superior performance. Taken together, these findings highlight the effectiveness of combining structured sampling, adaptive weighting, selective pressure, and controlled local search within Differential Evolution. The synergy of these components results in an optimizer that is not only faster but also remarkably stable across different problem types and dimensions.

A promising direction for future research is to explore how the proposed framework could be integrated with other well-established metaheuristic algorithms. Such a hybridization could leverage the strengths of different search strategies and potentially lead to more effective optimization performance. In addition, another interesting avenue is the incorporation of learning mechanisms such as reinforcement learning or adaptive parameter-learning techniques so that the algorithm can dynamically adjust its strategies and parameters based on the characteristics of the search landscape. Such a self-adaptive system could further enhance the stability, robustness, and overall efficiency of the optimization process.

Overall, this study demonstrates that carefully designed modifications to Differential Evolution can lead to substantial performance gains, and it sets the foundation for developing even more powerful and general-purpose optimization algorithms.

Author Contributions: G.K., V.C. and I.G.T. conceived of the idea and the methodology, and G.K. and V.C. implemented the corresponding software. G.K. conducted the experiments, employing objective functions as test cases, and provided the comparative experiments. I.G.T. performed the necessary statistical tests. All authors have read and agreed to the published version of the manuscript.

Funding: 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).

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

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

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

References

1. AutCarrizosa, E., Molero-Río, C., & Romero Morales, D. (2021). Mathematical optimization in classification and regression trees. Top, 29(1), 5-33.
2. Legat, B., Dowson, O., Garcia, J. D., & Lubin, M. (2022). MathOptInterface: a data structure for mathematical optimization problems. INFORMS Journal on Computing, 34(2), 672-689.
3. Su, H., Zhao, D., Heidari, A. A., Liu, L., Zhang, X., Mafarja, M., & Chen, H. (2023). RIME: A physics-based optimization. Neurocomputing, 532, 183-214.
4. Stilck França, D., & Garcia-Patron, R. (2021). Limitations of optimization algorithms on noisy quantum devices. Nature Physics, 17(11), 1221-1227.
5. Zhang, J., & Glezakou, V. A. (2021). Global optimization of chemical cluster structures: Methods, applications, and challenges. International Journal of Quantum Chemistry, 121(7), e26553.
6. Hu, Y., Zang, Z., Chen, D., Ma, X., Liang, Y., You, W., & Zhang, Z. (2022). Optimization and evaluation of SO2 emissions based on WRF-Chem and 3DVAR data assimilation. Remote Sensing, 14(1), 220.
7. Kaur, P., & Singh, R. K. (2023). A review on optimization techniques for medical image analysis. Concurrency and Computation: Practice and Experience, 35(1), e7443.
8. 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.
9. Wang, L., Cao, Q., Zhang, Z., Mirjalili, S., & Zhao, W. (2022). Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. Engineering Applications of Artificial Intelligence, 114, 105082.
10. 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.
11. 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.
12. Akintuyi, O. B. (2024). Adaptive AI in precision agriculture: a review: investigating the use of self-learning algorithms in optimizing farm operations based on real-time data. Research Journal of Multidisciplinary Studies, 7(02), 016-030.
13. Wang, Y., Ma, Y., Song, F., Ma, Y., Qi, C., Huang, F., ... & Zhang, F. (2020). Economic and efficient multi-objective operation optimization of integrated energy system considering electro-thermal demand response. Energy, 205, 118022.
14. 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.
15. Tang, K., Li, X., Suganthan, P. N., Yang, Z., & Weise, T. (2007). Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization. Nature inspired computation and applications laboratory, USTC, China, 24, 1-18.
16. Li, X., Tang, K., Omidvar, M. N., Yang, Z., Qin, K., & China, H. (2013). Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. gene, 7(33), 8.
17. Molina, D., & Herrera, F. (2015, May). Iterative hybridization of DE with local search for the CEC'2015 special session on large scale global optimization. In 2015 IEEE congress on evolutionary computation (CEC) (pp. 1974-1978). IEEE.
18. Vikhar, P. A. (2016, December). Evolutionary algorithms: A critical review and its future prospects. In 2016 International conference on global trends in signal processing, information computing and communication (ICGTSPICC) (pp. 261-265). IEEE.
19. Bartz-Beielstein, T., Branke, J., Mehnen, J., & Mersmann, O. (2014). Evolutionary algorithms. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 4(3), 178-195.
20. 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.
21. Pant, M., Zaheer, H., Garcia-Hernandez, L., & Abraham, A. (2020). Differential Evolution: A review of more than two decades of research. Engineering Applications of Artificial Intelligence, 90, 103479.
22. Sohail, A. (2023). Genetic algorithms in the fields of artificial intelligence and data sciences. Annals of Data Science, 10(4), 1007-1018.
23. Charilogis, V., Tsoulos, I. G., & Stavrou, V. N. (2023). An Intelligent Technique for Initial Distribution of Genetic Algorithms. Axioms, 12(10), 980.
24. van Rijn, S., Wang, H., van Leeuwen, M., & Bäck, T. (2016, December). Evolving the structure of evolution strategies. In 2016 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 1-8). IEEE.
25. Mezura-Montes, E., & Coello, C. A. C. (2008). An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. International Journal of General Systems, 37(4), 443-473.
26. Fogel, D. B. (1999). An overview of evolutionary programming. In Evolutionary algorithms (pp. 89-109). New York, NY: Springer New York.
27. Porto, V. W. (2018). Evolutionary programming. In Evolutionary Computation 1 (pp. 127-140). CRC Press.

28. Wang, H., Moon, I., Yang, S., & Wang, D. (2012). A memetic particle swarm optimization algorithm for multimodal optimization problems. Information Sciences, 197, 38-52.
29. Preuss, M. (2015). Multimodal optimization by means of evolutionary algorithms. Berlin/Heidelberg, Germany: Springer International Publishing.
30. Kennedy, J. (2006). Swarm intelligence. In Handbook of nature-inspired and innovative computing: integrating classical models with emerging technologies (pp. 187-219). Boston, MA: Springer US.
31. Parpinelli, R. S., & Lopes, H. S. (2011). New inspirations in swarm intelligence: a survey. International Journal of Bio-Inspired Computation, 3(1), 1-16.
32. 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.
33. Gad, A. G. (2022). Particle swarm optimization algorithm and its applications: a systematic review. Archives of computational methods in engineering, 29(5), 2531-2561.
34. 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.
35. Wu, L., Huang, X., Cui, J., Liu, C., & Xiao, W. (2023). Modified adaptive ant colony optimization algorithm and its application for solving path planning of mobile robot. Expert Systems with Applications, 215, 119410.
36. Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artificial intelligence review, 42, 21-57.
37. Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of global optimization, 39, 459-471.
38. Johari, N. F., Zain, A. M., Noorfa, M. H., & Udin, A. (2013). Firefly algorithm for optimization problem. Applied Mechanics and Materials, 421, 512-517.
39. Yang, X. S., & Slowik, A. (2020). Firefly algorithm. In Swarm intelligence algorithms (pp. 163-174). CRC Press.
40. Yang, X. S., & Hossein Gandomi, A. (2012). Bat algorithm: a novel approach for global engineering optimization. Engineering computations, 29(5), 464-483.
41. Yang, X. S. (2011). Bat algorithm for multi-objective optimisation. International Journal of Bio-Inspired Computation, 3(5), 267-274.
42. Storn, R., & Price, K. (1995). Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. International computer science institute.
43. 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.
44. Bai, Y., Wu, X., & Xia, A. (2021). An enhanced multi-objective differential evolution algorithm for dynamic environmental economic dispatch of power system with wind power. Energy Science & Engineering, 9(3), 316-329.
45. Penenko, A. V., Konopleva, V. S., & Penenko, V. V. (2022, May). Inverse modeling of atmospheric chemistry with a differential evolution solver: Inverse problem and Data assimilation. In IOP Conference Series: Earth and Environmental Science (Vol. 1023, No. 1, p. 012015). IOP Publishing.
46. Babanezhad, M., Behroyan, I., Nakhjiri, A. T., Marjani, A., Rezakazemi, M., & Shirazian, S. (2020). High-performance hybrid modeling chemical reactors using differential evolution based fuzzy inference system. Scientific Reports, 10(1), 21304.
47. Liu, L., Zhao, D., Yu, F., Heidari, A. A., Ru, J., Chen, H., ... & Pan, Z. (2021). Performance optimization of differential evolution with slime mould algorithm for multilevel breast cancer image segmentation. Computers in Biology and Medicine, 138, 104910
48. Balasubramanian, K., & Ananthamoorthy, N. P. (2021). Improved adaptive neuro-fuzzy inference system based on modified glowworm swarm and differential evolution optimization algorithm for medical diagnosis. Neural Computing and Applications, 33, 7649-7660.
49. 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.
50. Sofge, D., De Jong, K., & Schultz, A. (2002, May). A blended population approach to cooperative coevolution for decomposition of complex problems. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600) (Vol. 1, pp. 413-418). IEEE.
51. Van den Bergh, F., & Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization. IEEE transactions on evolutionary computation, 8(3), 225-239.
52. Gao, Y., & Wang, Y. J. (2007, August). A memetic differential evolutionary algorithm for high dimensional functions' optimization. In Third International Conference on Natural Computation (ICNC 2007) (Vol. 4, pp. 188-192). IEEE.
53. Chen, M., & Tan, Y. (2023). SF-FWA: A self-adaptive fast fireworks algorithm for effective large-scale optimization. Swarm and Evolutionary Computation, 80, 101314.

54. Sun, Y., & Cao, H. (2024). An agent-assisted heterogeneous learning swarm optimizer for large-scale optimization. Swarm and Evolutionary Computation, 89, 101627.
55. Wang, X., Wang, F., He, Q., & Guo, Y. (2024). A multi-swarm optimizer with a reinforcement learning mechanism for large-scale optimization. Swarm and Evolutionary Computation, 86, 101486.
56. Li, J. Y., Zhan, Z. H., Tan, K. C., & Zhang, J. (2022). Dual differential grouping: A more general decomposition method for large-scale optimization. IEEE Transactions on Cybernetics, 53(6), 3624-3638.
57. Li, J. Y., Du, K. J., Zhan, Z. H., Wang, H., & Zhang, J. (2022). Distributed differential evolution with adaptive resource allocation. IEEE transactions on cybernetics, 53(5), 2791-2804.
58. Charilogis, V., Tsoulos, I. G., Tzallas, A., & Karvounis, E. (2022). Modifications for the differential evolution algorithm. Symmetry, 14(3), 447.
59. Cheng, J., Zhang, G., & Neri, F. (2013). Enhancing distributed differential evolution with multicultural migration for global numerical optimization. Information Sciences, 247, 72-93.
60. Powell, M. J. D. (1989). A tolerant algorithm for linearly constrained optimization calculations. Mathematical Programming, 45, 547-566.
61. MacQueen, J. (1967, June). Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).
62. Sasmal, B., Hussien, A. G., Das, A., & Dhal, K. G. (2023). A comprehensive survey on aquila optimizer. Archives of Computational Methods in Engineering, 30(7), 4449-4476.
63. Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., & Gandomi, A. H. (2021). The arithmetic optimization algorithm. Computer methods in applied mechanics and engineering, 376, 113609.
64. SS, V. C. (2016). Smell detection agent based optimization algorithm. Journal of The Institution of Engineers (India): Series B, 97, 431-436.
65. Li, Y., & Wu, H. (2012). A clustering method based on K-means algorithm. Physics Procedia, 25, 1104-1109.
66. Arora, P., & Varshney, S. (2016). Analysis of k-means and k-medoids algorithm for big data. Procedia Computer Science, 78, 507-512.
67. Ali, M. M., & Kaelo, P. (2008). Improved particle swarm algorithms for global optimization. Applied mathematics and computation, 196(2), 578-593.
68. Koyuncu, H., & Ceylan, R. (2019). A PSO based approach: Scout particle swarm algorithm for continuous global optimization problems. Journal of Computational Design and Engineering, 6(2), 129-142.
69. 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.
70. 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.
71. Tsoulos, I.G., Charilogis, V., Kyrou, G., Stavrou, V.N. & Tzallas,A. (2025). OPTIMUS: A Multidimensional Global Optimization Package. Journal of Open Source Software, 10(108), 7584. Doi: https://doi.org/10.21105/joss.07584
72. Ge, H., Zhao, M., Hou, Y., Kai, Z., Sun, L., Tan, G., ... & Chen, C. P. (2020). Bi-space interactive cooperative coevolutionary algorithm for large scale black-box optimization. Applied Soft Computing, 97, 106798.
73. Hadi, A. A., Mohamed, A. W., & Jambi, K. M. (2019). LSHADE-SPA memetic framework for solving large-scale optimization problems. Complex & Intelligent Systems, 5(1), 25-40.
74. Molina, D., LaTorre, A., & Herrera, F. (2018, July). SHADE with iterative local search for large-scale global optimization. In 2018 IEEE congress on evolutionary computation (CEC) (pp. 1-8). IEEE.
75. 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.
76. Brodzicki, A., Piekarski, M., & Jaworek-Korjakowska, J. (2021). The whale optimization algorithm approach for deep neural networks. Sensors, 21(23), 8003
77. Luo, B., Su, X., Zhang, S., Yan, P., Liu, J., & Li, R. (2025). Analysis of a novel gas cycle cooler with large temperature glide for space cooling. Energy, 326, 136294.
78. Keerthika, R., Niranjan, S. P., & Komala Durga, B. (2025). A Survey on the tandem queueing models. Scope, 14, 134-148.