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 Differential Evolution algorithm (DE) is a population-based algorithm like Genetic Algorithms (GA) 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 The 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 algorithm; Evolutionary techniques; Stochastic methods; Large-scale problems

Received:
Revised:
Accepted:
Published:

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:

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

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], physics [2], chemistry [3], biology [5], medicine [4], agriculture [6] and economics [7]. 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 [8], 2013 [9] and 2015 [10], attracting the intense interest of the academic community.

To address these challenges, various heuristic and meta-heuristic approaches have been developed. Evolutionary Algorithms (EA) [11] 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 DE [12], GA [13], Evolutionary Strategies [14], Evolutionary Programming[15], Multimodal Optimization Algorithms [16]. Also, methods inspired by Swarm Intelligence [17] such as: Particle Swarm Optimization (PSO) [18], Ant Colony Optimization (ACO) [19], Artificial Bee Colony Optimization (ABC) [20], Firefly Algorithm (FA) [21], Bat Algorithm [22] are strong alternatives.

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 [23], 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 [24]. In chemistry, it has contributed to advances in atmospheric chemistry and the development of high-performance chemical reactors [25]. DE has also had significant implications in health-related areas, such as breast cancer research and medical diagnostics [26]. Despite its effectiveness, classical DE may face additional challenges when applied to high-dimensional and large-scale optimization problems. In such settings, the adaptation of control parameters becomes more demanding, while exploitation efficiency during later stages of the search process may be reduced. Moreover, as problem dimensionality increases, maintaining fast convergence and high solution accuracy can become more difficult. These observations have motivated extensive research efforts toward the development of enhanced DE variants, aiming to improve adaptability, scalability, and the balance between exploration and exploitation in large-scale optimization problems.

Recent studies have proposed various strategies to address large-scale optimization challenges, including cooperative coevolution [27], Particle Swarm Optimization [28], a DE approach [29], a self-adaptive Fast Fireworks Algorithm [30], swarm-based methods with learning mechanisms [31] and advanced decomposition techniques such as dual Differential Grouping [32]. These approaches highlight the ongoing research interest in designing efficient and scalable optimization frameworks for large-scale problems.

In this work, a unified and modular DE framework is proposed to improve efficiency, robustness, and convergence behavior in large-scale optimization problems. Instead of relying on a single evolutionary strategy or fixed parameter settings, the proposed approach integrates multiple enhancement mechanisms within a single DE framework. These mechanisms are designed to be independently configurable, allowing the algorithm to adapt more effectively to diverse problem characteristics while preserving the fundamental structure of classical DE.

The main contributions of this paper are summarized as follows:

- A unified and modular DE framework is introduced, integrating multiple control mechanisms within a single optimization scheme, including mutation weighting, parent selection, local refinement activation, and termination criteria.
- A k-means–based population sampling strategy is incorporated to preserve population structure and improve sampling efficiency in high-dimensional search spaces.
- Alternative mechanisms for computing the differential weight parameter are proposed, incorporating Number-based, Random, and Migrant strategies to enhance adaptability in large-scale optimization problems.
- An optional tournament-based parent selection strategy is employed to improve selection pressure while maintaining population diversity and robustness.
- A periodic local optimization refinement using deterministic local optimizers, such as BFGS, is integrated to enhance solution accuracy and accelerate convergence without compromising global exploration.
- A population-based termination criterion is introduced to enable early stopping when convergence stagnation is detected, significantly reducing unnecessary objective function evaluations.
- The proposed framework is specifically designed for large-scale global optimization problems and aims to achieve a more effective balance between exploration and exploitation compared to classical DE variants.

Although the individual enhancement mechanisms employed in the proposed framework, including adaptive termination strategies, clustering-based population initialization, adaptive differential weight schemes, and local optimization techniques, have been previously studied in the literature, their combined integration within a single DE framework is not straightforward. Each mechanism operates at a distinct stage of the evolutionary process and affects different aspects of the search dynamics, such as diversity preservation, convergence behavior, and computational efficiency.

The contribution of the present work lies in the systematic and modular coordination of these mechanisms within a unified optimization framework. By enabling controlled interaction between global exploration, adaptive parameter control, local exploitation, and termination mechanisms, the proposed approach facilitates synergistic effects that cannot be readily achieved through the isolated application of individual enhancements. This structured integration provides a principled framework for improving robustness and scalability in large-scale global optimization problems.

Importantly, the novelty of the proposed approach lies not in the introduction of new standalone operators, but in the principled way these mechanisms are coordinated within a single framework. Each component is explicitly designed to operate in synergy with the others, rather than being applied independently. In particular, structured sampling and adaptive differential weighting jointly influence population diversity and step-size control, while tournament selection ensures that informative population feedback is preserved for adaptive mechanisms to exploit. This coordinated interaction provides a theoretical justification for the combined framework and explains why its behavior cannot be reduced to a simple aggregation of existing techniques.

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. Differential Evolution Algorithm

2.1 The original Differential Evolution method

DE is a population-based evolutionary algorithm that has been widely used for continuous optimization problems. The method maintains a population of candidate solutions, which are iteratively evolved through the application of mutation, crossover, and selection operators. At each iteration, new candidate solutions are generated by combining information from multiple population members, while selection is performed based on objective function comparisons. The DE procedure begins by defining the population size NP. In order to ensure the feasibility of the classical DE mutation operator, which requires three distinct population members in addition to the target vector, a minimum population size of $NP \geq 4$ is required. In practice, the population size is often related to the dimensionality of the optimization problem. A commonly adopted guideline in the DE literature is to set $NP = 10n$, where n denotes the problem dimension, as this choice has been shown to provide robust performance across a wide range of problems without extensive parameter tuning. We note that alternative formulations relating the population size to the problem dimension have also been proposed in the literature; however, such choices are problem-dependent and do not affect the general applicability of the DE framework. The initial population is generated randomly within the search space and evaluated using the objective function. During each iteration, for every target vector $\mathbf{x}_i$, three distinct population members are randomly selected to construct a mutant vector through a differential mutation operation. This mutant vector is then combined with the target vector using a binomial crossover mechanism, producing a trial vector. If the trial vector achieves an objective function value that is not worse than that of the target vector, it replaces the target vector in the population. The evolutionary process continues until a termination criterion is satisfied, such as reaching a maximum number of iterations or meeting a convergence condition. The algorithm returns the best solution found during the search process. Regarding parameter settings, the crossover probability is set to $CR = 0.9$ and the differential weight to $F = 0.8$, following values commonly adopted in the DE literature [41]. These parameter values have been empirically shown to provide stable performance across a broad range of optimization problems without requiring problem-specific tuning. In this study, all parameters of the original DE algorithm are kept fixed throughout the experimental evaluation in order to ensure a fair and unbiased comparison with the proposed method. For clarity, the full steps of the original DE algorithm are summarized in Algorithm 1.

2.2 The proposed Differential Evolution method

The main contribution of the proposed approach is the formulation of a unified and modular DE framework that systematically integrates multiple control mechanisms within a single optimization scheme. Unlike most existing DE variants, which typically modify a single algorithmic component (e.g., mutation strategy or parameter adaptation), the proposed method allows the independent configuration and combined use of several algorithmic mechanisms while preserving the fundamental evolutionary structure of the original DE algorithm. The proposed method extends the classical DE framework by introducing a modular design that incorporates additional control components related to differential weight computation, parent selection, local refinement, and termination criteria. This design enables both the independent analysis of each component and their joint exploitation within a single, coherent optimization process, facilitating a systematic investigation of robustness and parameter sensitivity. The main methodological contributions of the proposed framework can be summarized as follows:

- Alternative mechanisms for differential weight computation

Algorithm 1 Original Differential Evolution Algorithm

```
INPUT
- $f$: objective function
- $NP$: population size
- $CR$: Crossover rate
- $F$: Differential weight
- $n$: Problem dimension
OUTPUT
-$x_{best}$
INITIALIZATION
-Generate an initial population of $NP$ candidate solutions$x_i,\ \ i = 1, \ldots, NP$, uniformly at random within the search bounds.
-Evaluate the objective function $f(x_i)$ for all individuals.
-Set $x_{best}$ as the individual with the best objective value.
main pseudocode
01 while stopping criterion is not met do
02     for each individual $i, i \in \{1..NP\}$ do
03         Select randomly three agents $a, b, c \in \{1..NP\}$
04         Generate mutant vector $u = a_j + F \times (b_j - c_j)$
05         Select a random index $R \in \{1..n\}$
06         for each dimension $j = 1$ to n do
07             Generate a random number $r_i \in [0, 1]$
08             if $r_j < CR\ or\ j = R$ then
09                 Set $y_j = u_j$.
10             else
11                 Set $y_j = x_{ij}$.
12             endif
13         endfor
14         if $f(y) \le f(x_i)$ then
15             Replace $x_i\ with\ y$
16         endif
17     endfor
18 endwhile
19 return $x_{best}$
```

- A k-means–based population sampling strategy is incorporated to preserve population structure and improve sampling efficiency in high-dimensional search spaces
- Optional tournament-based parent selection strategies,
- Periodic local refinement using a deterministic local optimizer
- A population-based termination criterion

The algorithm starts with an initialization phase in which a population of NP agents is randomly generated and evaluated using the objective function. Additional control parameters are also initialized at this stage, including the local search rate p_l, which determines the frequency of local refinement, the tournament size N_t, which controls selection pressure, the maximum number of generations N_g, the termination criteria N_I, and the iteration counter k. During the evolutionary process, different strategies for computing the differential weight F can be employed. These include a constant value, a random mechanism defined as $F = -0.5 + 2r$, where $r \in [0, 1]$, as well as a migrant-based strategy. For each agent, candidate solutions are generated through mutation and crossover operations, followed by a selection step based on objective function comparisons. In addition to the evolutionary operators, a deterministic local search procedure based on the BFGS method [36] may be periodically applied to refine promising solutions. The optimization process proceeds iteratively until a termination condition is satisfied, which may be defined either by a maximum number of generations or by a population-based convergence criterion. By jointly integrating stochastic variation, adaptive differential weight mechanisms, and deterministic local refinement within a unified framework, the proposed method aims to improve search efficiency and to achieve a more effective balance between exploration and exploitation. Moreover, the modular structure of the algorithm allows the systematic evaluation of the effect of individual algorithmic components, as demonstrated in the sensitivity analysis presented in Section 3. For clarity, the complete steps of the proposed DE algorithm are summarized in Algorithm 2.

The main steps of the proposed DE algorithm are illustrated in the flowchart presented in 1.

Algorithm 2 Proposed Algorithm

INPUT
- f: objective function
- NP: population size
- N_t: tournament size
- N_g: maximum number of iterations
- N_I: termination criteria
- CR: Crossover rate
- F: Differential weight
- n: Problem dimension
- k: iteration counter

OUTPUT
-x_{best}

INITIALIZATION
-Set as NP the population size (number of agents)
-Create randomly NP agents $x_i,\ \ i = 1, \ldots, NP$
-Compute the fitness value $f_i = f(x_i)$ for each agent
-Set as p_l the local search rate
-Set as N_g maximum number of iterations
-Set as N_I termination criteria
-Set as N_t tournament size
-Set $k \leftarrow 0$ as the iteration counter
-Set the parameter CR, with $CR \leq 1$
-Select the differential weight method F:
(a) Number : F is constant value.
(b) Random :$F = -0.5 + 2r, r \in [0, 1]$ poposed by Charilogis et al.[34].
(c) Migrant : migrant-based michanism[35].

main pseudocode

```
01 while stopping criterion is not met do
02     for each individual $i, i \in \{1..NP\}$ do
03         Select the agent $x_i$
04         Select randomly three distinct agents $x_a, x_b, x_c$
05         Choose a random integer $R \in [1, n]$
06         Create a trial point $x_t$
07         for $j \in \{1..n\}$ do
08             Select a random number $r \in [0, 1]$
09             if $r < CR\ or\ j = R$ then
10                 Set $x_{t,j} = x_{a,j} + F \times (x_{b,j} - x_{c,j})$
11             else
12                 Set $x_{t,j} = x_{ij}$
13             endif
14     endfor
15     Set $y_t = f(x_t)$
16     if $(y_t) \preceq f(x_i)$ then
17         Replace $x_i\ with\ x_t$
18     endif
19     Select a random number $r \in [0, 1]$
20     if $r \leq p_l$ then
21         Apply local search $x_i = LS(x_i)$[36]
22     endif
23         endfor
24         Set $k \leftarrow k + 1$
TERMINATION CHECK
25     if $k \geq N_g$ then terminate
26     Compute $\delta^{(k)} = \left| \sum_{i=1}^{NP} \mid f_i^{(k)} \mid - \sum_{i=1}^{NP} \mid f_i^{(k-1)} \mid \right|$
27     if $\delta^{(k)} \leq \varepsilon$ for $N_I$ iterations then terminate.
28 endwhile
29 return $x_{best}$
```

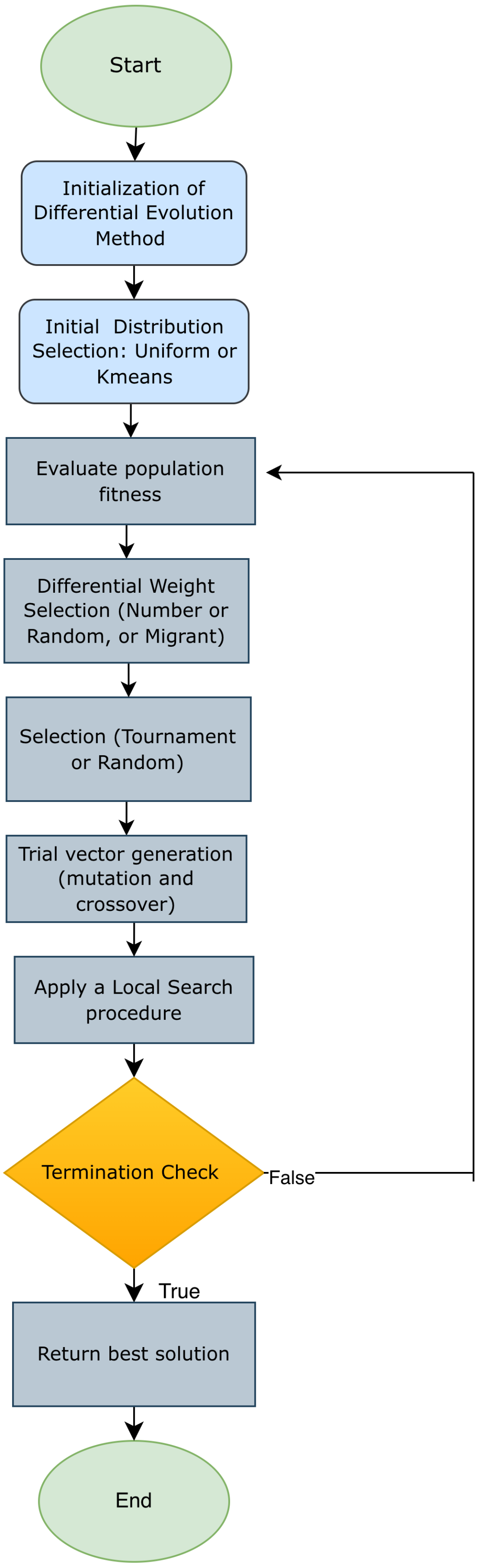
Start
Initialization of Differential Evolution Method
Initial Distribution Selection: Uniform or Kmeans
Evaluate population fitness
Differential Weight Selection (Number or Random, or Migrant)
Selection (Tournament or Random)
Trial vector generation (mutation and crossover)
Apply a Local Search procedure
Termination Check
False
True
Return best solution
End

Figure 1. The steps of the proposed DE algorithm.

Table 1. Benchmark test functions used in experimental study.

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
LLAGHER101	$f(\mathbf{x}) = \max_{i=1}^{101} \left[h_i - w_i \sqrt{\sum_{j=1}^{n} (x_j - c_{ij})^2}\right]$ $min : 100 + 1$	n	0
LLAGHER21	$f(\mathbf{x}) = \max_{i=1}^{21} \left[h_i - w_i \sqrt{\sum_{j=1}^{n} (x_j - c_{ij})^2}\right]$ $min : 10 + 10 + 1$	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

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 benchmark test functions were used in the conducted experiments. These functions have been widely adopted in previous studies [42–45]. In the present work, the test functions are evaluated using dimensionalities ranging from 25 to 150, where the constant n denotes the dimension of the objective function. The benchmark test functions used in the experimental study are summarized in Table 1, including their mathematical formulation, dimensionality, and global optimum values.

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 C++ using the freely available optimization environment of OPTIMUS [46], which can be downloaded from https://github.com/itsoulos/OPTIMUS. In addition to the proposed DE framework, a GA is employed as a baseline evolutionary method for comparative evaluation. The inclusion of GA provides a well-established reference approach in global optimization, allowing a clearer assessment of the performance of the proposed method. All comparative methods were independently implemented and evaluated within the same experimental framework and were not directly adopted from the corresponding literature. All algorithms were implemented in C++ and executed under identical hardware and software conditions to ensure a fair and consistent comparison. All algorithms were run using the same termination criterion, in order to ensure a fair and reproducible comparison. To guarantee comparability across all methods, common control parameters were fixed, as summarized in Table 2. Specifically, the number of agents (population size) was set to 200 for all algorithms, the maximum number of iterations was fixed at 200, and the local search rate was uniformly set to 0.05. These shared settings ensure that performance differences arise from algorithmic design rather than differences in experimental configuration. Algorithm-specific parameters were selected according to standard practices reported in the literature and were kept constant throughout all experiments. No problem-specific parameter tuning was applied in order to avoid bias and to maintain methodological consistency. The parameter settings for both the proposed method and the GA are summarized in the Table 2

Table 2. 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	0.05 (5%)
-	Selection Rate for	0.05 (5%)
-	Selection method for	Roulette

The parameter values selected for the GA were chosen based on standard practices in evolutionary computation and preliminary empirical evaluation. All GA parameters were kept fixed throughout the experimental study to avoid problem-specific tuning and to ensure a fair and consistent comparison with the proposed method.

3.3. 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 DE 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
LLAGHER21_25	2261(0.90)	3815(0.90)	6364(0.90)
LLAGHER21_50	4503(0.50)	5277(0.50)	9643(0.50)
LLAGHER21_100	1756(0.53)	1523(0.53)	1521(0.53)
LLAGHER21_150	1662(0.27)	1521(0.27)	1526(0.27)
LLAGHER101_25	2769(0.90)	3472(0.90)	5657(0.90)
LLAGHER101_50	4890(0.50)	6950(0.50)	7454(0.50)
LLAGHER101_100	5886(0.53)	6846(0.53)	9505(0.53)
LLAGHER101_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 2 presents a pairwise statistical comparison of the three migration strategies MIGRANT (T), NUMBER (T), and RANDOM (T)with respect to the required number of function evaluations. The Kruskal–Wallis test ($p = 0.3$) does not indicate the presence of statistically significant overall differences among the three strategies, suggesting comparable distributions of computational cost. This outcome is consistent with the results of the pairwise t-tests, for which none of the comparisons reach statistical significance ($p > 0.05$). Accordingly, the observed differences in medians and dispersion across the strategies are not supported as statistically significant. The MIGRANT (T) strategy exhibits a slightly lower average number of function evaluations; however, this difference is not accompanied by statistical significance and remains within the range of stochastic variability. Overall, the

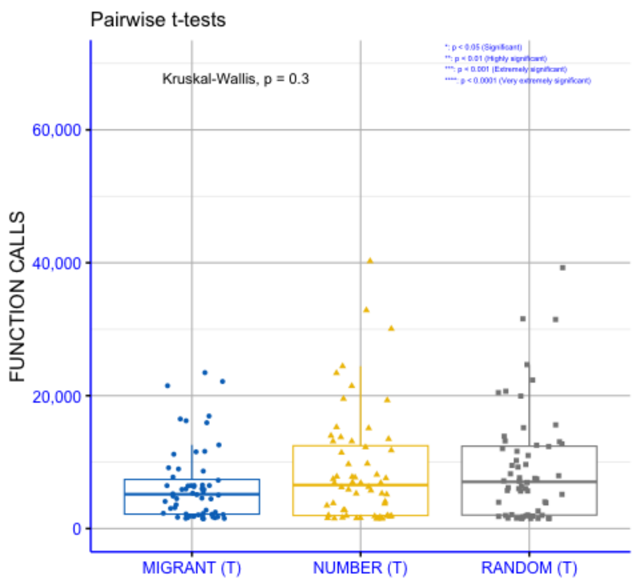

Figure 2. A statistical comparison of the proposed with different weight selection.

results indicate similar behavior of the three migration strategies in terms of computational cost.

3.4. *The effect of selection mechanism*

Table 4 compares 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
LLAGHER21_25	2261(0.90)	5412(0.90)	6364(0.90)	2891(0.90)
LLAGHER21_50	4503(0.50)	11988(0.50)	9643(0.50)	3311(0.50)
LLAGHER21_100	1756(0.53)	1565(0.53)	1521(0.53)	1524(0.53)
LLAGHER21_150	1662(0.27)	1490(0.27)	1526(0.27)	1520(0.27)
LLAGHER101_25	2769(0.90)	5180(0.90)	5657(0.90)	3016(0.90)
LLAGHER101_50	4890(0.50)	21179(0.50)	7454(0.50)	4878(0.50)
LLAGHER101_100	5886(0.53)	26739(0.53)	9505(0.53)	4507(0.53)
LLAGHER101_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 3 presents a pairwise statistical comparison of the four strategies MIGRANT (T), MIGRANT (R), RANDOM (T) and RANDOM (R) based on their distributions of function calls. The Kruskal–Wallis test indicates a statistically significant overall difference among the groups (p = 0.0019), suggesting that at least one strategy differs from the others in terms of computational cost. To further examine these differences, pairwise t-tests were conducted, with p-values annotated using standard significance notation (ns: $p > 0.05$, : $p < 0.05$, *: $p < 0.01$, ***: $p < 0.001$, ****: $p < 0.0001$) The pairwise analysis shows that MIGRANT (T) differs significantly from MIGRANT (R) and RANDOM (R), with the corresponding comparisons reaching higher levels of statistical significance. The strategies MIGRANT (R) and RANDOM (T) exhibit intermediate behavior, with several pairwise comparisons indicating statistically significant differences at moderate significance levels. Comparisons labeled as "ns" indicate pairs for which no statistically significant differences are detected. Overall, the distributions of function calls indicate lower evaluation counts for the

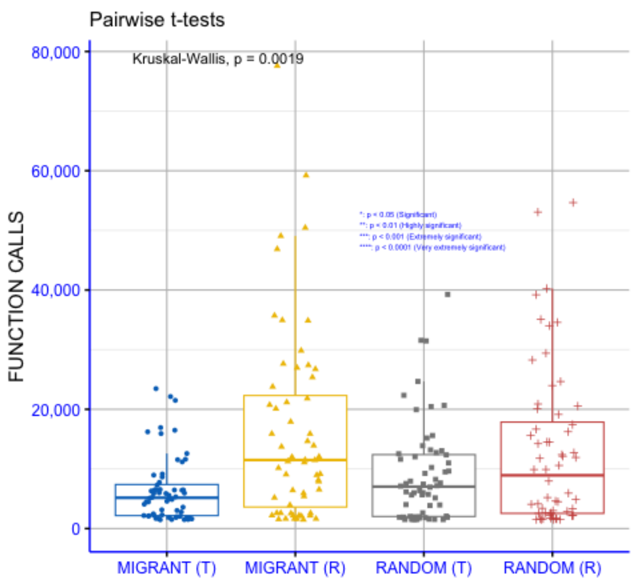

Figure 3. Statistical Comparison of Random and Tournament Selection Strategies on Optimization Performance.

MIGRANT-based strategies, particularly MIGRANT (T), relative to the RANDOM-based variants. These results describe differences in computational cost among the examined strategies under the considered experimental conditions.

3.5. *The effect of sampling method*

In Table 5 tournament selection is used to choose the samples that participate in the core operator of DE. Four strategies for computing the differential weight are evaluated: random weight with uniform sampling (Random(U)), random weight with k-means sampling[39,40] (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[52] 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 1738 for 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 DE 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
LLAGHER21_25	2261(0.90)	6920(0.03)	6364(0.90)	34112(0.03)
LLAGHER21_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)

Table 4 presents the pairwise t-test statistical comparison of the four strategies MIGRANT (K), RANDOM (K), MIGRANT (U) and RANDOM (U) based on their distributions of function calls. The Kruskal–Wallis test indicates a statistically significant overall difference among the groups (p = 0.034), suggesting that at least one strategy differs from the

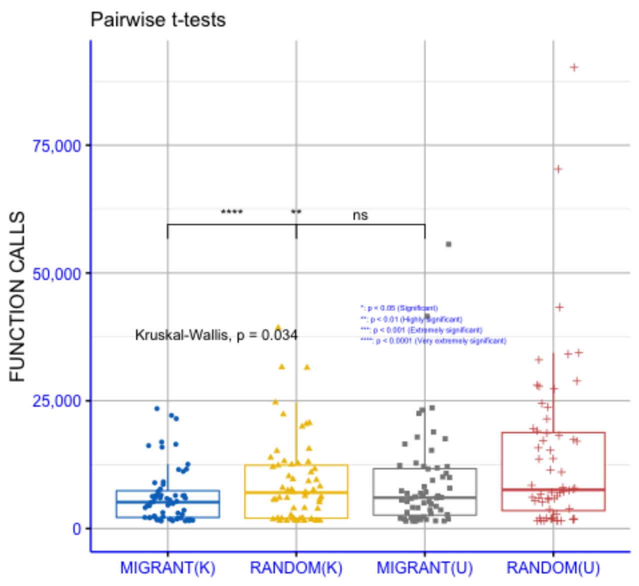

Figure 4. Statistical Comparison of Different Sampling Method Combinations in DE Performance.

others in terms of computational cost. To further examine these differences, pairwise t-tests were conducted, with p-values annotated using conventional significance notation (ns: p > 0.05, : p < 0.05, *: p < 0.01, ***: p < 0.001, ****: p < 0.0001). The pairwise comparisons indicate that MIGRANT (K) differs significantly from RANDOM (K) and from MIGRANT (U), while no statistically significant difference is observed between MIGRANT (U) and RANDOM (U). Comparisons labeled as "ns" suggest statistically indistinguishable behavior between the corresponding strategy pairs. Overall, the distributions of function calls show lower evaluation counts for MIGRANT (K) relative to some of the other strategies under the examined conditions. The U-type variants exhibit similar behavior to the RANDOM (U) baseline. These results describe differences in computational cost among the considered strategies, without implying uniform or dominant superiority across all comparisons.

3.6. 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 DE.

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 5 presents pairwise statistical comparisons among the four parameter configurations (0.001, 0.01, 0.03, and 0.05) based on their distributions of function evaluations. The global Kruskal–Wallis test indicates a statistically significant overall difference among the groups (p = 2.9e−08), suggesting that the choice of parameter value is associated with differences in computational cost. Pairwise comparisons were conducted using independent t-tests, with p-values interpreted according to conventional significance notation (ns: p >

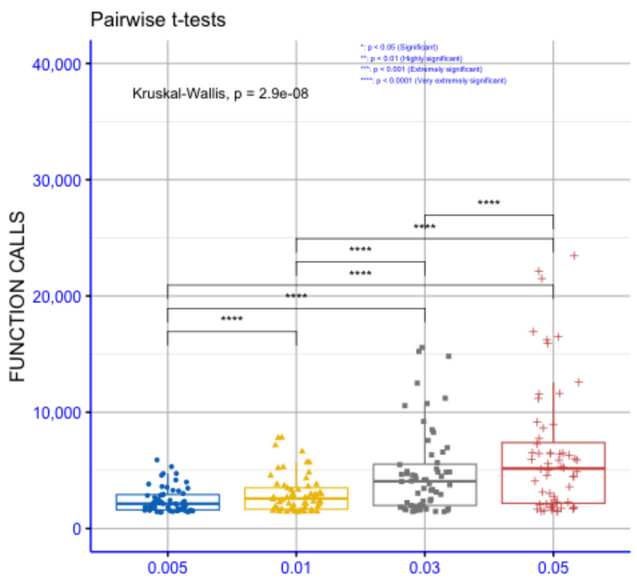

Figure 5. Statistical comparison for the proposed method and different values of parameter p_l.

0.05, : p < 0.05, *: p < 0.01, ***: p < 0.001, ****: p < 0.0001). As shown in the figure, a substantial number of pairwise comparisons reach statistically significant levels, while a smaller subset does not exhibit statistically significant differences. This indicates that, for many comparisons, the examined parameter configurations are associated with distinguishable behavior in terms of the number of function calls. Overall, the results indicate that different parameter settings correspond to varying distributions of function evaluations under the considered experimental conditions. The observed differences suggest a systematic influence of the parameter choice on optimizer behavior, without implying uniform or absolute superiority of a single configuration across all comparisons.

3.7 Parameter Sensitivity Analysis

To assess the robustness of the proposed method with respect to its main control parameters, a sensitivity analysis was conducted focusing on the local search rate p_l, the population size NP, and the tournament size N_t. Each parameter was varied independently, while all remaining parameters were fixed to their default values. For each configuration, the algorithm was executed multiple times under identical experimental conditions, and the average number of function calls was recorded.

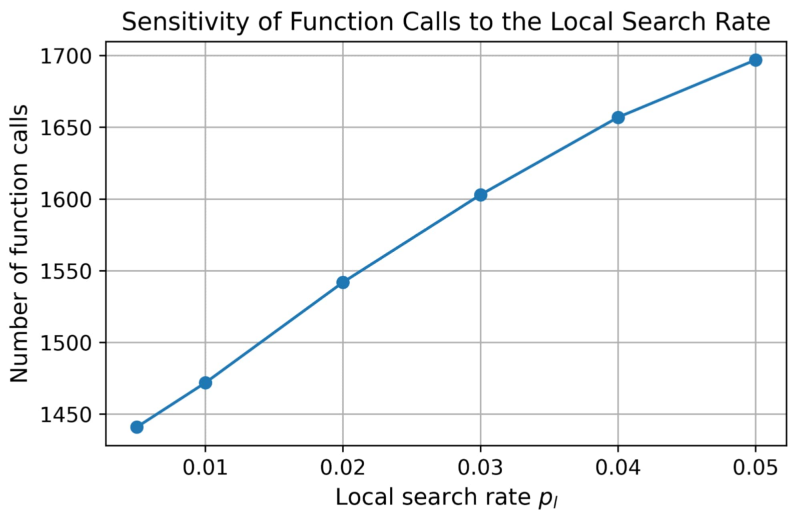

Figure 6. Sensitivity of function calls to the Local Search

Table 6 illustrates the sensitivity of the number of function calls with respect to the local search ratep_l. The results indicate a gradual increase in computational cost as $p_l$increases, which can be attributed to the more frequent activation of the local refinement procedure. Nevertheless, the observed variation is smooth, and no abrupt performance degradation is observed for moderate changes in the local search rate. 438–442

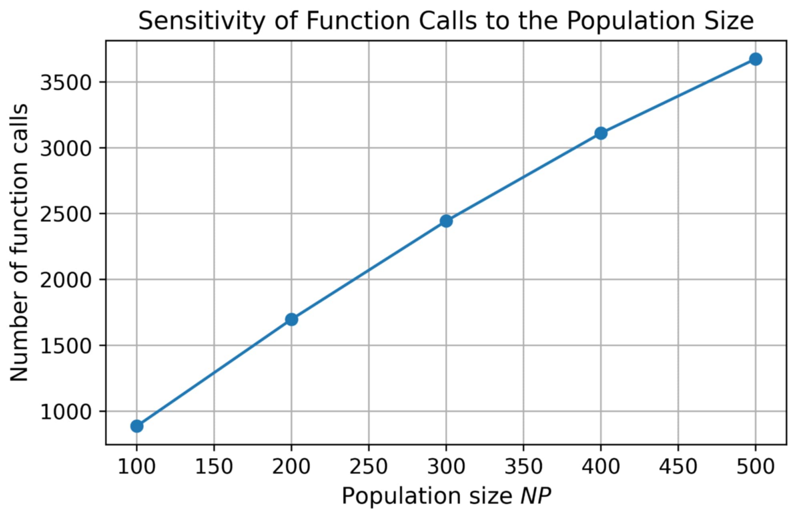

Figure 7. Sensitivity of function calls to the Population Size

Table 7 presents the effect of the population size NP on the number of function calls. As expected, increasing the population size leads to a higher computational cost due to the larger number of individuals evaluated at each iteration. However, the trend remains monotonic and predictable, indicating stable scaling behavior rather than sensitivity to a particular population size. 443–447

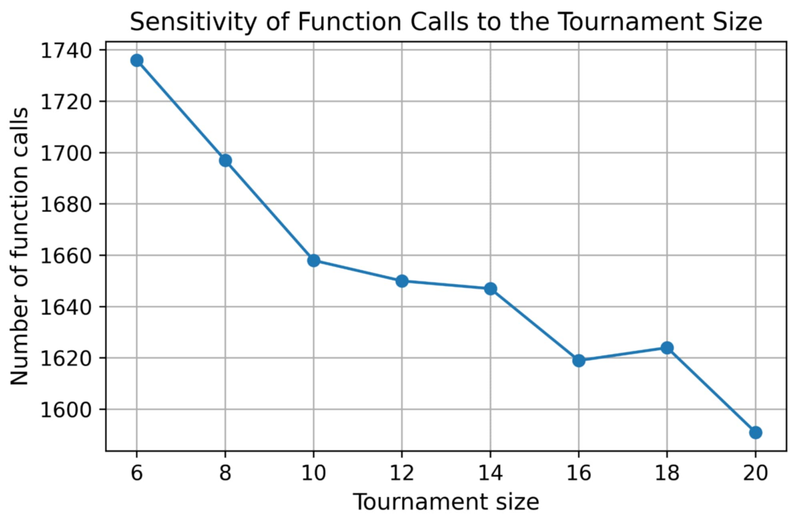

Figure 8. Sensitivity of function calls to the Tournament Size

Table 8 shows the sensitivity of the algorithm to the tournament size N_t when tournament-based parent selection is employed. The results demonstrate that moderate values of N_t yield comparable performance, while larger tournament sizes slightly reduce the number of function calls by increasing selection pressure. Importantly, the performance differences across the tested range remain limited, suggesting that the algorithm is not highly sensitive to the precise choice of N_t .

Overall, the sensitivity analysis indicates that the proposed method exhibits stable behavior across a broad range of parameter values. Performance variations are gradual and mainly occur at extreme settings, suggesting that the method can be transferred to different problem instances without requiring extensive parameter tuning. Regarding the termination limits, no sensitivity analysis was performed, as the same termination criterion was applied uniformly to all experiments and all reference functions.

3.8 The proposed method in comparison with others

Table 7 presents the results of a comparative analysis of various optimization methods (BICCA[47], MLSHADESPA[48], SHADE_ILS[49], Differential Evolution (DE)[12], Genetic Algorithm (GA)[13], Whale Optimization Algorithm (WOA)[50,51], Particle Swarm Optimization (IPSO)[18], 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 7. 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)

Table 9 illustrates the distributions of function evaluations for the PROPOSED optimizer and the baseline algorithms. The Kruskal–Wallis test indicates a statistically sig-

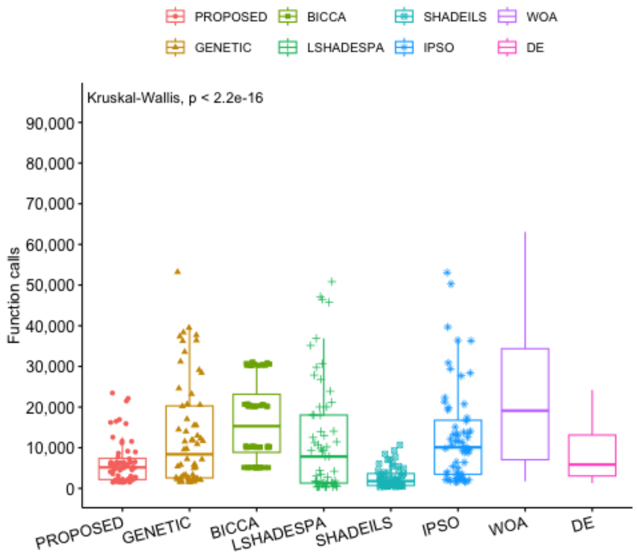

Figure 9. A statistical comparison of the proposed with other optimization methods.

nificant overall difference among the methods ($p < 2.2e-16$), suggesting that at least one optimizer differs from the others with respect to the distribution of function-call counts. As shown in the figure, the PROPOSED method is associated with lower median values and reduced dispersion in the number of function evaluations compared with the baseline algorithms. These differences in the distributions indicate that the PROPOSED optimizer exhibits distinct computational behavior under the examined experimental conditions. Overall, the statistical analysis and the observed distributions suggest that the PROPOSED method requires fewer function evaluations relative to the considered baselines. The results describe systematic differences in computational cost among the optimizers, without attributing the observed behavior to a single dominant factor.

To further investigate the scalability of the proposed optimization framework, additional experiments were conducted in very high dimensions and more specifically for dimensions 200,300,600,1100.

Table 8. Experimental results using different very high-dimensional optimization methods.

FUNCTION	WOA	BICCA	SHADE_ILS	MLSHADESPA	DE	PROPOSED	GA	IPSO
ELLIPSOIDAL_200	49353	40915	9635	35675	27985	22123	31875	30853
ELLIPSOIDAL_300	51152	61333	9903	43473	43569	39328	52414	53185
ELLIPSOIDAL_600	80978	122625	21797	85276	73034	70020	78370	81032
ELLIPSOIDAL_1100	124378	223787	25607	123451	84912	80815	114632	113982

The results of these additional experiments are reported in Table 8. As the dimension increases, the proposed method is associated with consistently lower numbers of objective function evaluations compared with the baseline algorithms, indicating favorable scalability characteristics in very high-dimensional optimization scenarios.

Table 9. Experimental results using mean and standard deviation.

FUNCTION	WOA	BICCA	SHADE_ILS	MLSHADESPA
RASTRIGIN_25	8.42(26.07)	45.36(44.73)	8.55(26.28)	6.26(19.39)
RASTRIGIN_50	27.79(47.44)	86.06(23.96)	52.16(54.44)	44.27(45.56)
RASTRIGIN_100	19.07(58.51)	258.95(63.67)	70.24(80.80)	51.24(55.91)
RASTRIGIN_150	12.17(46.32)	312.08(68.05)	128.81(90.00)	102.81(64.24)
STEPELLIPSOIDAL_25	1.18(6.48)	2107.69(2049.63)	387.30(1193.21)	387.30(1193.21)
STEPELLIPSOIDAL_50	0(0)	6975.21(1772.99)	3876.67(4034.57)	3876.67(4034.57)
STEPELLIPSOIDAL_100	0(0)	10197.95(1242.43)	5232.05(5786.36)	5232.05(5786.36)
STEPELLIPSOIDAL_150	0(0)	9769.58(1069.59)	10796.25(6776.41)	10796.25(6776.41)

Table 10. Experimental results using mean and standard deviation.

FUNCTION	DE	PROPOSED	GA	IPSO
RASTRIGIN_25	1.49(4.60)	2.88(8.8)	5.33(16.87)	0.36(1.34)
RASTRIGIN_50	23.71(24.60)	19.96(21.33)	51.30(52.68)	8.42(0.09)
RASTRIGIN_100	55.12(61.30)	28.92(32.80)	62.51(68.96)	15.02(16.65)
RASTRIGIN_150	132.52(83.51)	65.10(41.07)	126.06(79.14)	35.71(22.56)
STEPELLIPSOIDAL_25	0(0)	36.38(131.59)	143.05(443.83)	2.39(7.74)
STEPELLIPSOIDAL_50	0(0)	625.24(709.06)	3076.57(3214.39)	129.48(159.90)
STEPELLIPSOIDAL_100	0(0)	1124.16(1293.24)	5163.19(5722.02)	379.90(481.85)
STEPELLIPSOIDAL_150	24.60(16.03)	2791.42(1817.21)	10676.31(6679.24)	1007.36(777.37)

In addition to the success rates, Tables 9 and 10 present the mean value and standard deviation of the objective function across 30 independent runs, in order to provide a more informative comparison between the methods, especially in cases where the success rates are the same or lower.

3.9 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 [53] and the Tandem Queueing System [54]. 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**
  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 GasCycle scenario presents a more computationally demanding optimization problem, allowing a clearer assessment of algorithmic scalability under increased complexity.

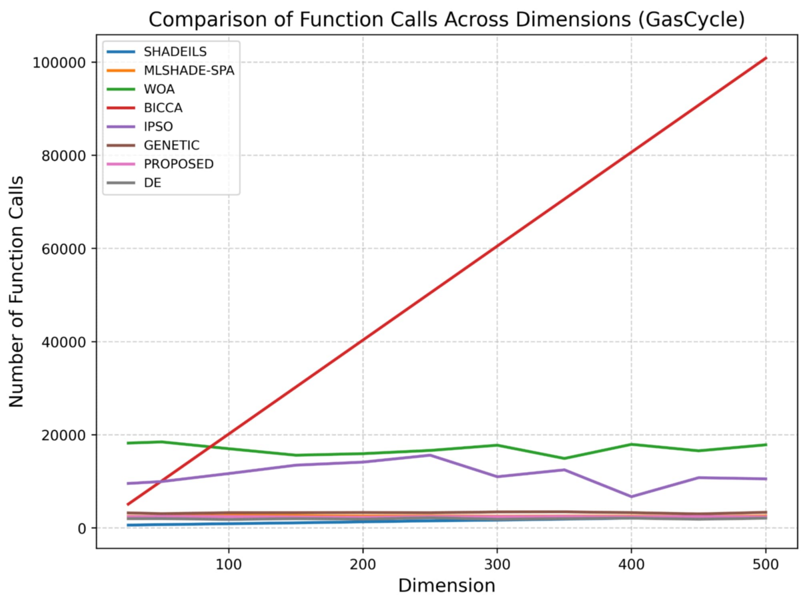

Figure 10. Comparison of Function Calls Across Dimension (GasCycle)

For GasCycle, as illustrated in 10, the proposed algorithm maintains a stable and competitive number of function calls across all dimensions. Compared to methods that show pronounced growth in evaluations at higher dimensions, the proposed approach exhibits a more controlled increase, indicating effective adaptation to the structure of the GasCycle problem. This behavior suggests that the method can efficiently utilize function evaluations without excessive computational overhead in large-scale cases. 538–544

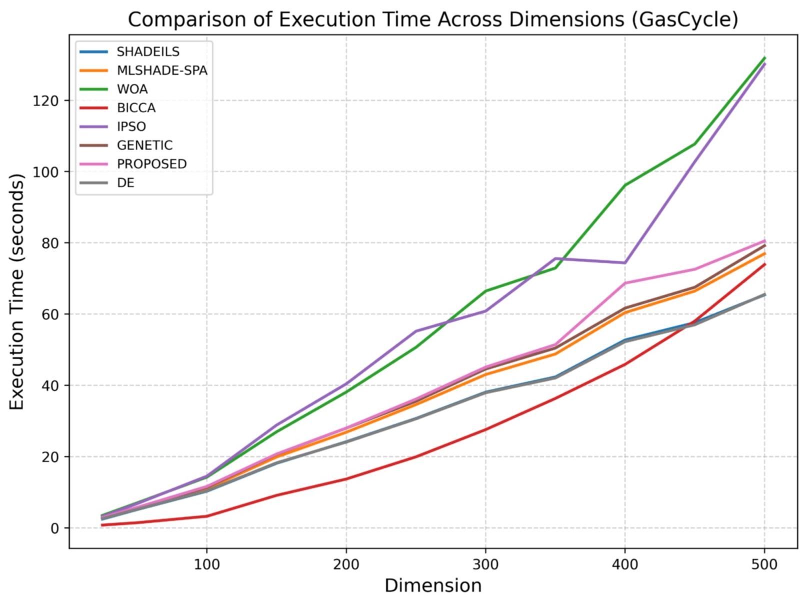

Figure 11. Comparison of Execution Time Across Dimension (GasCycle)

The execution time analysis for GasCycle, shown in 11, aligns closely with the function call results. The proposed algorithm achieves a balanced runtime profile, with execution time increasing smoothly as dimensionality grows. In contrast to approaches that suffer from substantial runtime escalation, the proposed method maintains reasonable computational demands even at high dimensions, highlighting its suitability for complex, large-scale optimization tasks. 545–550

- **Tandem Space Trajectory** (MGA-1DSM, EVEEJ + 2×Saturn) 551

Vars **(D=18):** $\boldsymbol{x} = [t_0, T_1, T_2, T_3, T_4, T_{5A}, T_{5B}, s_1, s_2, s_3, s_4, s_{5A}, s_{5B}, r_p, k_{A1}, k_{A2}, k_{B1}, k_{B2}]^\top$. 552

$$\begin{aligned} &7000 \le t_0 \le 10000, \\ &30 \le T_1 \le 500,\ 30 \le T_2 \le 600,\ 30 \le T_3 \le 1200, \\ &30 \le T_4 \le 1600,\ 30 \le T_{5A}, T_{5B} \le 2000, \\ &0 \le s_{1..4}, s_{5A}, s_{5B}, r_p, k_{A1}, k_{A2}, k_{B1}, k_{B2} \le 1. \end{aligned}$$ 553

Objective: 554

$$\min_{\boldsymbol{x}} \ \Delta V_{\text{tot}} = \Delta V_{\text{launch}}(T_1) + \Delta V_{\text{legs}}(T_1{:}T_4) + \Delta V_A + \Delta V_B + \Delta V_{\text{DSM}}(\boldsymbol{s}, r_p) - G_{\text{GA}} - G_J + P_{\text{hard}} + P_{\text{soft}},$$ 555

556

$$P_{\text{soft}} = \beta \max\Big\{0,\ (T_1 + \cdots + T_4 + \tfrac{1}{2}(T_{5A} + T_{5B})) - 3500\Big\}.$$ 557

Notes: ΔV_{launch} decreases (log-like) in T_1 (≥ 6 km/s floor), leg/branch costs decrease with TOF. 558

The figures corresponding to the Tandem scenario illustrate the behavior of the evaluated algorithms in terms of function calls and execution time as the problem dimension increases. As expected, higher dimensionality leads to increased computational effort for all methods; however, notable differences in scalability can be observed. 559–562

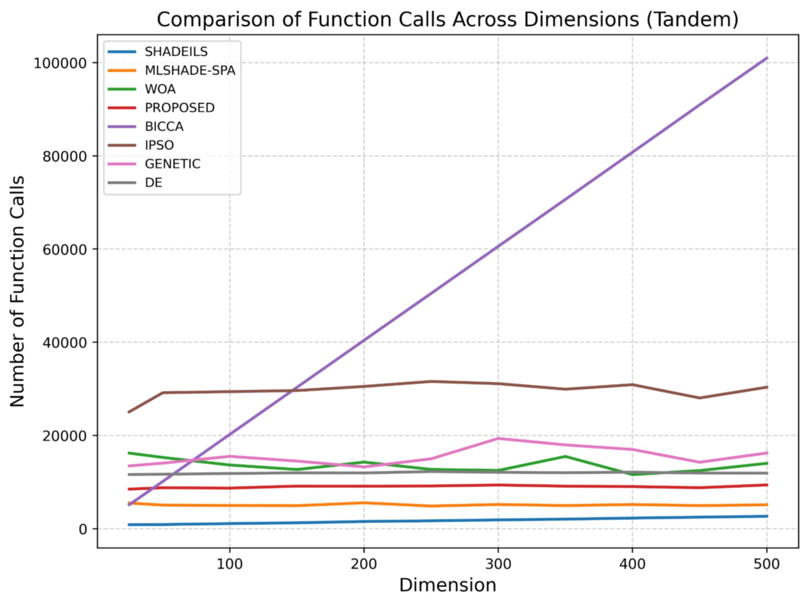

Figure 12. Comparison of Function Calls Across Dimension (Tandem)

In the Tandem case, as shown in Figure 12, the proposed algorithm demonstrates stable and consistent behavior across all tested dimensions, maintaining a relatively low number of function calls. Its performance remains competitive with the most efficient approaches and is clearly more scalable than methods such as GA, BICCA, and IPSO, which exhibit a rapid increase in function calls as dimensionality grows. The controlled growth observed for the proposed method indicates effective search dynamics and an appropriate balance between exploration and exploitation in large-scale settings. 563–570

The execution time results for the Tandem scenario, presented in Figure 13, further confirm these observations. The proposed algorithm shows smooth and predictable scaling with increasing problem dimension, avoiding the steep runtime growth observed in more computationally demanding methods. Although execution time naturally increases for larger dimensions, the rate of increase remains moderate, sug- 571–575 gesting that the internal computational cost of the proposed approach is well managed and suitable for practical large-scale applications in the Tandem scenario.

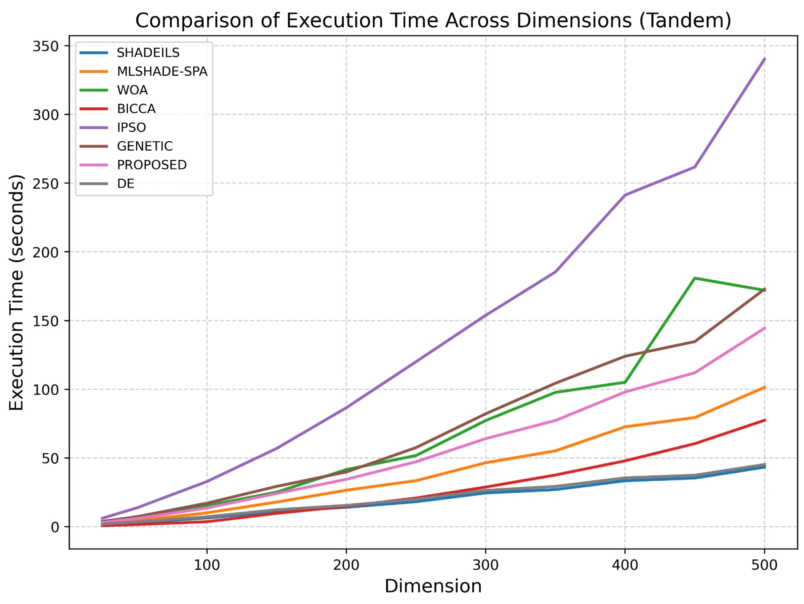

Figure 13. Comparison of Execution Time Across Dimensions (Tandem)

Across both Tandem and GasCycle scenarios, the proposed algorithm demonstrates consistent scalability in terms of both function evaluations and execution time. Its stable behavior under increasing dimensionality indicates that it represents a reliable and efficient alternative for large-scale optimization problems, without incurring excessive computational cost.

4. Discussion

The experimental results provide important insights into how design choices in sampling, parameter adaptation, selection pressure, and local refinement collectively shape the behavior of Differential Evolution (DE). Rather than acting independently, these components interact in ways that significantly influence convergence efficiency and stability, a phenomenon already highlighted in foundational studies on DE that emphasize the role of mutation–selection interplay and parameter control in stochastic optimization processes [23,41]. When such interactions are properly structured and coordinated, the algorithm exhibits more reliable convergence behavior and reduced sensitivity to random effects. In large-scale optimization settings, where the dimensionality of the search space increases the complexity of variable interactions and amplifies stochastic disturbances, the importance of these interactions becomes even more pronounced. The results observed in this study suggest that careful coordination of DE components can mitigate instability and uncontrolled performance degradation, extending the principles established in classical DE formulations to more challenging large-scale scenarios.

A key observation concerns the role of sampling strategies. The use of k-means clustering to guide population sampling consistently leads to lower computational cost and reduced variance compared to uniform sampling. This behavior can be attributed to the preservation of population structure, which helps prevent redundant exploration and encourages a more balanced coverage of promising regions of the search space. The effectiveness of k-means in producing compact and representative partitions of the search space has been well established in the clustering literature [39,52]. In the context of evolutionary optimization, our experimental results indicate that leveraging such structured sampling becomes particularly beneficial in large-scale settings, where purely random sampling often leads to inefficient exploration. The tighter distributions observed for configurations employing k-means indicate improved stability, an essential property for large-scale optimization.

The differential weight mechanism further reinforces this structured behavior. Among the examined strategies, the MIGRANT-based approach demonstrates consistently lower function evaluation requirements while maintaining identical success rates compared to static or random schemes. This suggests that exploiting feedback from the evolving population enables the algorithm to adapt its step sizes more effectively, leading to more economical progress through the search space. Such observations are in line with existing literature on adaptive parameter control in DE, where learning-based or feedback-driven strategies are known to outperform fixed parameter choices, especially as problem dimensionality increases [12].

Selection pressure also plays a critical role in shaping the overall dynamics of the search. Tournament selection, when combined with adaptive differential weighting, provides a controlled bias toward high-quality solutions without excessively reducing population diversity. In contrast, random selection introduces unnecessary stochasticity that can disrupt the learning process, particularly when coupled with adaptive mechanisms that rely on informative population feedback. The results indicate that even moderate selection pressure can significantly improve the reliability of adaptive strategies by ensuring that useful information is retained and exploited across generations, as also discussed in the evolutionary computation literature [15,41].

The influence of local search frequency highlights the importance of moderation in hybrid metaheuristics. While local refinement can enhance solution quality, excessive application increases computational overhead and may interfere with global exploration. The experimental findings show that a very low local search rate achieves the best trade-off between exploitation and efficiency, supporting the view that local search should act as a complementary mechanism rather than a dominant driver of the optimization process. This observation aligns with prior work on memetic and hybrid evolutionary algorithms, where controlled and infrequent local search is often more effective than aggressive refinement [10].

When considered collectively, these observations explain the strong and consistent performance of the proposed method relative to classical and modern optimizers. Rather than relying on aggressive parameter settings or complex hybridization, the method benefits from a balanced integration of structured exploration and adaptive exploitation. The comparative results indicate that the proposed approach achieves competitive or superior performance with lower variability, suggesting improved robustness across diverse problem landscapes. Importantly, where statistical tests do not indicate significant pairwise differences, the observed performance trends are interpreted as consistent empirical behavior rather than strict dominance.

Overall, the discussion highlights that the performance gains achieved by the proposed framework stem from thoughtful algorithmic structure rather than brute-force complexity. By guiding the search through informed sampling, adaptive weighting, and controlled selection pressure, the algorithm avoids both premature convergence and inefficient exploration. These findings reinforce a broader principle in large-scale evolutionary optimization: intelligent structure and feedback-driven adaptation are often more effective than increased randomness or parameter proliferation, in agreement with large-scale benchmark studies and methodological guidelines [8,45].

As is common in population-based metaheuristic methods, no formal convergence or complexity guarantees are provided for the proposed approach. Instead, scalability and computational behavior are assessed empirically through extensive experimental evaluation, following standard practice in large-scale optimization research [8,9]. The inclusion of large-scale benchmark functions and real-world engineering problems, such as the Tandem and GasCycle scenarios, enables a practical assessment of convergence trends and computational cost under increasing dimensionality. The observed behavior in terms of function evaluations and execution time indicates controlled scalability in large-scale optimization settings.

5. Conclusions

This work explored large-scale optimization through a systematically enhanced version of the DE 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 GA, 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 DE. 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 DE 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. 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.
2. 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.
3. 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.
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. Li, X. (2024). Optimization of crop tissue culture technology and its impact on biomolecular characteristics. Molecular & Cellular Biomechanics, 21(2).
6. Xiao, L., Wang, G., Wang, E., Liu, S., Chang, J., Zhang, P., ... & Luo, Z. (2024). Spatiotemporal co-optimization of agricultural management practices towards climate-smart crop production. Nature Food, 5(1), 59-71.
7. Hassan, M. H., Kamel, S., Jurado, F., & Desideri, U. (2024). Global optimization of economic load dispatch in large scale power systems using an enhanced social network search algorithm. International Journal of Electrical Power & Energy Systems, 156, 109719.
8. 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.
9. 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.
10. 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.
11. Li, P., Hao, J., Tang, H., Fu, X., Zhen, Y., & Tang, K. (2024). Bridging evolutionary algorithms and reinforcement learning: A comprehensive survey on hybrid algorithms. IEEE Transactions on evolutionary computation.
12. 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.
13. Charilogis, V., Tsoulos, I. G., & Stavrou, V. N. (2023). An Intelligent Technique for Initial Distribution of Genetic Algorithms. Axioms, 12(10), 980.
14. Lange, R., Tian, Y., & Tang, Y. (2024, July). Large language models as evolution strategies. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 579-582).
15. Cicirello, V. A. (2024). Evolutionary computation: Theories, techniques, and applications. Applied Sciences, 14(6), 2542.
16. Cheng, S., Wang, X., Zhang, M., Lei, X., Lu, H., & Shi, Y. (2024). Solving multimodal optimization problems by a knowledge-driven brain storm optimization algorithm. Applied Soft Computing, 150, 111105.

17. Kong, L. S., Jasser, M. B., Ajibade, S. S. M., & Mohamed, A. W. (2024). A systematic review on software reliability prediction via swarm intelligence algorithms. Journal of King Saud University-Computer and Information Sciences, 36(7), 102132.
18. 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.
19. 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.
20. Ibrahim, A. O., Elfadel, E. M. E., Hashem, I. A. T., Syed, H. J., Ismail, M. A., Osman, A. H., & Ahmed, A. (2025). The Artificial Bee Colony Algorithm: A Comprehensive Survey of Variants, Modifications, Applications, Developments, and Opportunities. Archives of Computational Methods in Engineering, 1-35.
21. Singh, A. K., & Kumar, A. (2025). Multi-objective: hybrid particle swarm optimization with firefly algorithm for feature selection with Leaky ReLU. Discover Artificial Intelligence, 5(1), 192.
22. Dao, T. K., & Nguyen, T. T. (2025). A review of the bat algorithm and its varieties for industrial applications. Journal of Intelligent Manufacturing, 36(8), 5327-5349.
23. Storn, R., & Price, K. (1995). Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. International computer science institute.
24. 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.
25. 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.
26. 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
27. Yao, X., & Chong, S. Y. (2025). Cooperative Coevolution for Large-Scale Optimization. In Coevolutionary Computation and Its Applications (pp. 199-270). Singapore: Springer Nature Singapore.
28. McGovarin, Z., Engelbrecht, A. P., & Ombuki-Berman, B. M. (2024). Stochastic Grouping and Subspace-Based Initialization in Decomposition and Merging Cooperative Particle Swarm Optimization for Large-Scale Optimization Problems. In Canadian AI.
29. Yue, X., Liao, Y., Peng, H., Kang, L., & Zeng, Y. (2025). A high-dimensional feature selection algorithm via fast dimensionality reduction and multi-objective differential evolution. Swarm and Evolutionary Computation, 94, 101899.
30. Chen, M., & Tan, Y. (2023). SF-FWA: A self-adaptive fast fireworks algorithm for effective large-scale optimization. Swarm and Evolutionary Computation, 80, 101314.
31. Sun, Y., & Cao, H. (2024). An agent-assisted heterogeneous learning swarm optimizer for large-scale optimization. Swarm and Evolutionary Computation, 89, 101627.
32. 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.
33. 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.
34. Charilogis, V., Tsoulos, I. G., Tzallas, A., & Karvounis, E. (2022). Modifications for the differential evolution algorithm. Symmetry, 14(3), 447.
35. Cheng, J., Zhang, G., & Neri, F. (2013). Enhancing distributed differential evolution with multicultural migration for global numerical optimization. Information Sciences, 247, 72-93.
36. Powell, M. J. D. (1989). A tolerant algorithm for linearly constrained optimization calculations. Mathematical Programming, 45, 547-566.
37. 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.
38. Sulaiman, A. T., Bello-Salau, H., Onumanyi, A. J., Mu'azu, M. B., Adedokun, E. A., Salawudeen, A. T., & Adekale, A. D. (2024). A particle swarm and smell agent-based hybrid algorithm for enhanced optimization. Algorithms, 17(2), 53.
39. Li, Y., & Wu, H. (2012). A clustering method based on K-means algorithm. Physics Procedia, 25, 1104-1109.
40. Arora, P., & Varshney, S. (2016). Analysis of k-means and k-medoids algorithm for big data. Procedia Computer Science, 78, 507-512.
41. Price, K. V., Storn, R. M., & Lampinen, J. A. (2005). Differential evolution: a practical approach to global optimization. Berlin, Heidelberg: Springer Berlin Heidelberg.
42. Ali, M. M., & Kaelo, P. (2008). Improved particle swarm algorithms for global optimization. Applied mathematics and computation, 196(2), 578-593.
43. 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.

44. 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.
45. 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.
46. 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
47. 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.
48. 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.
49. 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.
50. 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.
51. Brodzicki, A., Piekarski, M., & Jaworek-Korjakowska, J. (2021). The whale optimization algorithm approach for deep neural networks. Sensors, 21(23), 8003
52. 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).
53. 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.
54. Keerthika, R., Niranjan, S. P., & Komala Durga, B. (2025). A Survey on the tandem queueing models. Scope, 14, 134-148.