Article

The Echo Optimizer: A Novel Metaheuristic Inspired by Acoustic Reflection Principles

Vasileios Charilogis[1], Ioannis G. Tsoulos[2,*]

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

Abstract: The Echo Optimizer Method is an innovative optimization technique inspired by the natural behavior of sound echoes. It is based on generating modified solutions (echoes) that combine directed reflection toward the best-known solution and random noise that attenuates over time. The method introduces two groundbreaking mechanisms to enhance performance: an approximate evaluation system that avoids costly computations for unpromising solutions, and an echo memory that stores and reuses past evaluations. These mechanisms enable a significant reduction in computational resources (up to 40–60% fewer evaluations) while maintaining the method's effectiveness. The Echo Optimizer excels in balancing exploration of the solution space with exploitation of the best-known solutions, demonstrating impressive performance in problems with numerous local minima and high dimensionality. Experimental tests on standard optimization problems have shown faster convergence and reduced result variability compared to classical methods, making it a highly attractive choice for various optimization challenges, particularly in cases where evaluating the objective function is computationally expensive.

Keywords: Optimization; Echo Optimizer; Evolutionary Algorithms; Global Optimization; Adaptive Termination; Mutation Strategies; Metaheuristics;

Citation: Charilogis, V.; Tsoulos, I.G. The Echo Optimizer: A Novel Metaheuristic Inspired by Acoustic Reflection Principles. *Journal Not Specified* **2024**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

1. Introduction

Global optimization deals with finding the absolute lowest point (global minimum) of a continuous objective function $f(x)$ defined over a bounded, n-dimensional search space S. Mathematically, the goal is to identify the point x^* in S where $f(x)$ attains its smallest possible value:

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

where:

- $f(x)$ is the objective function to minimize (e.g., cost, error, or energy).
- S is a compact (closed and bounded) subset of R^n, often defined as an n-dimensional hyperrectangle:

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

Here, a_i and b_i are the lower and upper bounds for each variablex_i, confining the search to a specific region.

Optimization constitutes a central field of computational mathematics with applications to multifaceted scientific and industrial problems. Optimization methods are classified into broad categories according to their underlying strategies and the properties of the problems they address. Among the most well-known techniques are classical gradient methods such as steepest descent [1] and Newton's method [2], stochastic methods like Monte Carlo [3] algorithms and simulated annealing [4,5] population-based methods including genetic algorithms [6] and differential evolution [7–10], convex optimization methods such as the ellipsoid method [11,12] and cutting-plane method [13], simplex-based methods like the Nelder-Mead method [14], response surface methods including krigin [15], trust-region methods like Bayesian optimization [16,17], conjugate gradient methods such as Fletcher-Reeves [18] and Polak-Ribière [19,20], constrained optimization methods including penalty function methods and interior-point methods, decomposition approaches like Benders decomposition [21,22] and Dantzig-Wolfe decomposition [23], space-partitioning methods such as DIRECT [24] and branch-and-bound [25,26], neural network-based methods including reinforcement learning algorithms, socially-inspired methods like particle swarm optimization [27,28] and ant colony optimization [29], physics-inspired methods including crystal structure optimization [30] and gravitational search algorithms [31], hybrid methods such as neuro-fuzzy algorithms [32], and biologically-inspired methods like photosynthesis algorithms [33] and DNA-based computing [34].

Within this context, the Echo Optimizer method (EO) introduces a novel approach based on the physical analogy of sound echoes. The core concept involves generating modified solutions that simulate sound reflection, with systematic adjustment of exploration intensity through a decay factor. This technique combines elements from population-based methods and stochastic techniques, offering a flexible mechanism for addressing diverse optimization problems. The method's ability to balance solution space exploration with exploitation of known optimal solutions makes it particularly effective for high-dimensional problems and non-convex functions.

The presentation of the method will focus on its theoretical foundations, algorithmic design, and experimental performance compared to other contemporary techniques. Furthermore, we will analyze its application potential to real-world problems, along with the challenges arising from its use in complex systems. This work aims to highlight the method's distinctive properties that make it a valuable addition to the toolkit of modern optimization techniques.

The rest of the paper is organized as follows:

The introduction provides the background and motivation for the study. The Echo Optimizer method is presented in Section 2, detailing the core algorithm and its components. Subsection 2.2 introduces the technique of approximate evaluations, while Subsection 2.1 describes the technique of echo memory. Followed by subsection 2.3 where the complete algorithm is described. The experimental setup and benchmark results are discussed in Section 3, which includes an overview of the benchmark functions in Subsection 3.1 and the experimental results in Subsection 3.2. Finally, the conclusions are presented in Section 4, summarizing the key findings and implications of the study.

2. The Echo Optimizer method

The EO algorithm is an evolutionary method inspired by the natural behavior of sound echoes, combining mathematical optimization strategies with physical analogies. Its core principle lies in generating solutions (called "echoes") through a directed reflection of current solutions towards the best-known solution, coupled with the addition of random noise that attenuates over time. The initial population of solutions is generated randomly within the bounds of the search space, and each solution is evaluated based on an objective function. During each iteration, solutions are updated using a combination of directed reflection, which pulls them closer to the best-known solution, and random noise, which allows exploration of new regions in the solution space. The noise diminishes as the algorithm progresses, enabling a transition from exploration to exploitation. This mechanism ensures the algorithm's ability to balance the discovery of new potential solutions with the refinement of promising ones, making it effective for tackling high-dimensional problems and complex landscapes with multiple local minima.

The basic EO algorithm generates new solutions, called echoes, based on a combination of directed reflection and random noise.

For each dimension d of the solution x_i, the updated value $echo_d$ is computed as:

$$echo_d = x_{i,d} + coef \cdot (x_{best,d} - x_{i,d}) + noise_d \tag{2}$$

Where:

- $coef \cdot (x_{best,d} - x_{i,d})$: A directed reflection term that moves the solution toward the best-known solution x_{best}.
- $coef$: Reflection factor
- $noise_d = random[-1, 1] \cdot (1 - currentDelay)$: A random noise term that attenuates as $currentDelay$ decreases over iterations.
- $currentDelay = initDelay - (initDelay - finalDelay) \cdot (\frac{iter}{iter_{max}})$: A decay factor that adjusts over the course of $iter_{max}$ iterations.

2.1. The echo memory technique

The memory technique introduces a unique mechanism to the EO algorithm by enabling the storage and reuse of information from previously evaluated solutions, significantly reducing computational overhead. Specifically, each evaluated solution is stored in an echo memory along with its corresponding objective function value. When a new solution is generated, the algorithm checks if a similar solution exists in memory, using a predefined tolerance level to determine similarity. If a match is found, the stored value is retrieved instead of recalculating the objective function. This feature is particularly beneficial in problems where evaluating solutions is computationally expensive. Moreover, the memory is dynamically updated with new, improved solutions, ensuring the algorithm's adaptability to evolving optimization scenarios. This mechanism enhances the algorithm's efficiency by focusing computational resources on exploring truly novel areas of the search space, rather than redundantly evaluating similar solutions.

In the memory-enhanced EO configuration, the algorithm stores previously evaluated solutions and their corresponding fitness values to avoid redundant computations. When a new solution echo is generated, the algorithm checks if a similar solution exists in the memory within a predefined tolerance (e: small number e.g. 0.5):

$$|echo_d - memory_{j,d}| < e \tag{3}$$

If a match is found:

- Retrieve $memoryFitness_j$.
- If $memoryFitness_j < f_i$, update the current solution and its fitness:
  - $x_i = echo$
  - $f_i = memoryFitness_j$

Similarity in the EO method is determined exclusively by the Euclidean distance between the coordinates of the candidate solution and those stored in memory. Similarity is evaluated by comparing the differences in the coordinate values for each dimension. If the absolute difference between the respective values is smaller than the predefined tolerance, the solution is considered similar to one already stored in memory. The calculation is based on the Equation 3, where the result is compared against the tolerance value e (Table 1). The objective function values are not used in the similarity determination process. The evaluation relies solely on geometric proximity, specifically the position of the solutions within the search space.

2.2. The approximate evaluations technique

The approximation technique allows the algorithm to estimate the quality of solutions quickly and efficiently, avoiding unnecessary computations for solutions that are unlikely to provide improvements. When a new solution is generated, instead of immediately evaluating it using the objective function, the algorithm applies an approximation function to predict its quality based on its proximity to the best-known solution. If the approximation exceeds a predefined threshold, the solution is rejected without further processing. This selective evaluation process accelerates convergence by focusing on promising solutions and reducing the number of expensive objective function calculations. The approximation technique thus improves the algorithm's speed without sacrificing accuracy or its ability to explore the solution space, making it particularly efficient in scenarios where objective function evaluations are resource-intensive.

In the configuration with approximate evaluations, the algorithm uses a heuristic to estimate the quality of a solution before computing the objective function. The approximate fitness is calculated as:

$$approxFitness = \left(\sum_{d=1}^{n} (echo_d - x_{best,d})^2 \right) \cdot currentDelay \tag{4}$$

If: $approxFitness > approxThreshold \cdot f_i$:

then the solution is rejected without evaluating the objective function, and the algorithm proceeds to the next individual.

The memory and approximation techniques in the EO differ from similar approaches such as Long Term Memory Assistance for Evolutionary Algorithms (LTMA) [35] and Surrogate Modelling [36], both in philosophy and implementation. In the case of memory, EO utilizes a dynamic storage structure that contains previously evaluated solutions and their fitness values. This mechanism avoids redundant computations by comparing new solutions with those already stored in memory and using the stored fitness value if the new solution is sufficiently similar. In contrast, LTMA maintains historical data from previous generations of evolutionary algorithms and focuses on strategically guiding the optimization process based on long-term trends and changes in the population of solutions. EO's memory mechanism is more localized and aimed at reducing computational costs, while LTMA operates on a broader scale, emphasizing exploration guidance. Regarding approximation techniques, EO relies on heuristic estimates to discard unpromising solutions before the objective function is evaluated. These estimates are based on the distance of the new solution from the best-known solution and a decay factor that adjusts over the iterations. On the other hand, Surrogate Modelling creates accurate mathematical or statistical models, such as Gaussian Processes, to approximate the objective function. These models require training on previously evaluated data points and are used to select the most promising solutions. EO employs a lighter approach suitable for rapid execution, whereas Surrogate Modeling offers higher accuracy but comes with greater computational cost. Overall, EO emphasizes simplicity and reduced computational overhead, while LTMA and Surrogate Modelling pursue strategic guidance and increased accuracy, respectively, often at the expense of higher resource and time requirements.

2.3. The overall algorithm

The overall algorithm of the method follows:

Algorithm 1 Pseudocode of EO

Algorithm: Echo Optimization with Approximate Evaluations and Memory
Input:
- NP: Population size
- $initDelay \in [0,1]$: Initial delay factor
- $finalDelay \in [0,1]$: Final delay factor
- $reflectionFactor \in [0,1]$: Reflection coefficient
- $iter_{max}$: Maximum iterations
- $approxThreshold$: Approximation threshold (e.g., 0.2 - 20%)
- $maxMemorySize$: Memory capacity
- $technique$ ∈{ECHO, ECHO+APP, ECHO+MEM, ECHO+APP+MEM}: Variant selection
- $searchSpace$: Feasible solution bounds

Output:
- x_{best}: Optimal solution found, f_{best}: Corresponding fitness value

Initialization:
Initialize population $X = \{x_i | x_i\ U(searchSpace), i = 1, ..., NP\}$
Evaluate initial fitness $F = \{f = f(x_i) | i = 1, ..., NP\}$
Set (x_{best}, f_{best}) = $argmin_{(x_i, f_i)} f_i$
Initialize empty memory: $M = \varnothing$, $F_M = \varnothing$

Main Optimization Loop:
for iter = 1 to $iter_{max}$ do
    $currentDelay = initDelay - (initDelay - finalDelay) \cdot (\frac{iter}{iter_{max}})$
    for each individual x_i in X do
        // Echo vector generation
        for d = 1 to dim do
            $echo_d = x_{i,d} + coef \cdot (x_{best,d} - x_{i,d}) + U(-1,1) \cdot (1 - currentDelay)$
            $echo_d = clamp(echo_d, searchSpace.lower_d, searchSpace.upper_d)$
        end for
        // Approximation evaluation (Optional)
        if $technique$ = "ECHO+APP" then
            $approxFitness = \left(\sum_{d=1}^{n} (echo_d - x_{best,d})^2\right) \cdot currentDelay$
            if $approxFitness > approxThreshold \cdot f_i$ then
                continue to next individual
            end if
        end if
        // Memory lookup (Optional)
        if $technique$ = "ECHO+MEM" then
            if $\exists j : ||echo - M_j|| < \varepsilon$ then
                if $F_{M_j} < f_i$ then
                    $x_i \leftarrow echo$
                    $f_i \leftarrow F_{M_j}$
                    UpdateBest(x_i,f_i)
                end if
                continue to next individual
            end if
        end if
        // Full evaluation
        $f_{echo} = f(echo)$
        if $f_{echo} \leq f_i$ then
            $x_i \leftarrow echo$
            $f_i \leftarrow f_{echo}$
            UpdateBest(x_i,f_i)
            // Update memory (if applicable)
            if $technique$ = "memory" and $|M| < maxMemorySize$ then
                $M \leftarrow M \cup \{echo\}$
                $F_M \leftarrow F_M \cup \{f_{echo}\}$
            end if
        end if
    end for
    // Local search phase (Optional)[37]
    for each individual x_i in X do
        if $U(0,1) < localSearchRate$ then
            $x_{refined}, f_{refined} = localSearch(x_i)$
            if $f_{refined} < f_i$ then
                $x_i \leftarrow x_{refined}$
                $f_i \leftarrow f_{refined}$
                UpdateBest(x_i,f_i)
            end if
        end if
    end for
    if termination criteria SR is meet then end for1
end for
return (x_{best}, f_{best})

The EO algorithm (Algorithm 1) begins by initializing its parameters, which include the population size NP, the initial delay factor $initDelay$, the final delay factor $finalDelay$, the reflection factor $coef$, the maximum number of iterations $iter_{max}$, the approximation threshold $approxThreshold$, and the maximum memory size $maxMemorySize$. A random population of solutions x_i is generated within the predefined bounds of the search space, where i ranges from 1 to NP. Each solution in the population is evaluated using the objective function to compute its fitness f_i. The best solution, referred to as x_{best}, and its fitness f_{best} are identified. An echo memory, echoMemory, is initialized as an empty structure, along with its corresponding memory fitness values.

The optimization process enters the main loop, which runs for a maximum of $iter_{max}$ iterations. At each iteration, the delay factor $currentDecay$ is computed as a linear interpolation between $initDelay$ and $finalDelay$, depending on the current iteration $iter$ relative to $iter_{max}$. For each solution x_i in the population, a new solution (echo) is generated dimension by dimension. The new value for each dimension $echo_d$ is calculated by adding three components: the reflection term, which directs the solution toward $x_{best,d}$, the noise term, which introduces randomness and decreases as $currentDecay$ reduces over time, and the original value of the dimension. Boundary correction is applied to ensure the new solution remains within valid limits.

After generating the echo, optional techniques may be applied to enhance computational efficiency. If the approximation technique is enabled, an approximate fitness $approxFitness$ is calculated based on the proximity of the echo to x_{best}. If $approxFitness$ exceeds $approxThreshold \cdot f_i$, the echo is discarded without further evaluation, and the algorithm moves to the next solution. If the memory technique is enabled, the algorithm checks whether the echo exists in the echo memory within a tolerance. If a match is found, the stored fitness value is used. If the stored fitness is better than the current f_i , x_i and f_i are updated accordingly. For echoes that pass these checks or if the optional techniques are not applied, the fitness of the echo $echoFitness$ is calculated by evaluating the objective function. If $echoFitness$ is better than f_i , the echo replaces the current solution x_i, and f_i is updated. If the echo is the best solution found so far, x_{best} and f_{best} are updated.

An optional local search phase may follow, where individual solutions are refined with a certain probability. If the refined fitness improves upon the current fitness, the solution and its fitness are updated. The algorithm continues to iterate until $iter_{max}$ is reached or a termination condition is met, at which point the best solution x_{best} is returned as the result of the optimization process.

2.4. Parameters Sensitivity

The reflection factor ($coef$), which regulates the attraction toward the current best solution, directly affects the algorithm's ability to escape local minima. Values around 0.5 generally work well across various problems, but for functions with steep gradients or many small local minima, fine-tuning may be necessary. In contrast, the parameters that control the linear decay exhibit greater stability, with default values providing a balanced transition from exploration to exploitation without causing major disruptions.

The approximation threshold ($approxThreshold$) determines the tendency to reject candidate solutions and directly influences convergence speed. Values near 0.2 provide an acceptable balance between accuracy and speed. However, in problems with wide basins of attraction, it can be increased up to 0.3 to accelerate the process, whereas in problems with sharp ridges, it may be reduced to 0.1 to avoid mistakenly discarding valid solutions. The local search rate may also require attention, especially in problems with numerous local minima, as higher values tend to improve reliability at the cost of increased computational overhead.

For practical applications, it is recommended to begin with the default parameter values, followed by gradual adjustments depending on the problem type. Avoiding extreme values in sensitive parameters is important, as values outside reasonable bounds can significantly degrade performance. Empirical testing on representative problems remains the most reliable strategy for parameter tuning, while the development of automatic parameter control mechanisms would be a valuable extension in future versions of the algorithm.

The memory tolerance parameter (e) is particularly sensitive and directly affects the efficiency of the memory mechanism. The default value of e = 0.5 works well for many problems, typically corresponding to 5–10% of the variable range in standard optimization benchmarks. However, the optimal value depends on the characteristics of the objective function. Small values (e < 0.2) greatly increase the number of stored solutions, improving accuracy but also leading to excessive memory usage, increased computational cost during similarity checks, and a risk of overfitting to local variations. On the other hand, large values (e > 1.0) reduce the effectiveness of memory by allowing false associations between dissimilar solutions, lowering reuse rates, and leading to redundant objective function evaluations. For functions with steep slopes or narrow valleys (e.g., ROSENBROCK), lower values (e = 0.2–0.4) are recommended to ensure accuracy. For functions with wide attraction basins (e.g., SPHERE), higher values (e = 0.6–0.8) improve reuse efficiency. In noisy functions, setting e $\leq$ 0.3 helps avoid contamination of the memory with unstable solutions. The interaction between e and the memory size (maxMemorySize) is also critical. When memory is limited, a slightly increased e improves coverage without overloading the system, while in cases with large memory availability, a smaller e allows more precise mapping of the search space. There is no universally optimal value; selection must rely on experimental evaluation across representative subproblems.

3. Experimental setup and benchmark results

This section first introduces the benchmark functions selected for experimental evaluation, followed by a comprehensive analysis of the conducted experiments. The study systematically examines the various parameters of the ECHO+APP+MEM algorithm to assess its reliability and effectiveness in different optimization scenarios. The complete parameter configurations used throughout these experiments are documented in Table 1.

Table 1. Parameters and settings

PARAMETER	VALUE	EXPLANATION
NP	200, 50, $4 + \lfloor 3 \cdot \log(\text{dimension}) \rfloor$	Population
$iter_{max}$	200	Maximum number of iterations for all methods
$initialDelay$	0.9*	Initial echo delay factor for EO
$finalDelay$	0.1*	Final echo delay factor for EO
$coef$	0.5*	Reflection factor towards best solution for EO
$approxThreshold$	0.2 (20%)*	Approximate threshold for EO
e	0.5**	Echo memory tolerance for EO
SR	$\delta_{sim}^{(iter)} = \left\| f_{sim,min}^{(iter)} - f_{sim,min}^{(iter-1)} \right\|$ [38–40]	Stopping rule for all methods
N_s	12	Similarity $count_{max}$ for stopping rule
LS	0.02 (2%), Tables: 3, 4, 5	Local search rate
T_s	Tourament size 8 [41]	Selection of GA
C_{rate}	double, 0.1 (10%) (classic values)	Crossover for GA
M_{rate}	double, 0,05 (5%) (classic values)	Mutation for GA
c_1, c_2	1.193	Cognitive and Social coefficient for PSO
w	0.721	Inertia for PSO
c_1, c_2	1.494	Cognitive and Social coefficient for LCPSO
w	0,729	Inertia for LCPSO
F	0.5	Initial scaling factor for SaDE
CR	0.5	Initial crossover rate for SaDE
P_r	-0.301, $\text{memory}_{\text{tolerance}} = (10)^{-0.301} \approx 0.5$	Precision control parameter for LTMA

*The parameter values of the EO method were selected to ensure the method's maximum possible effectiveness.

**The method uses a tolerance of 0.5 to determine if a new solution is similar to those stored in memory, providing a balance between sensitivity and flexibility. This value allows small deviations in variables while preserving the ability to distinguish significant changes. In problems with a range of values from -5 to 5, the tolerance represents about 5–10% of the total range, preventing excessive storage of solutions or overly strict comparisons that could reduce efficiency. The choice is based on experimental analysis to balance reducing unnecessary objective function calls with maintaining solution quality. A tolerance of 0.5 meets typical accuracy requirements by detecting meaningful improvements while ignoring minor fluctuations. For problems with different characteristics, the tolerance can be adjusted, but it serves as a good initial choice for most applications.

3.1. *Test Functions*

The experiments were conducted on a wide range of test functions [5,42,43] as shown in Table 2. The functions are standard benchmark optimization test functions, which are widely used in the literature and have not been modified with shifting or rotation for the experimental tests.

Table 2. The benchmark functions used in the conducted experiments

NAME	FORMULA	DIMENSION	$Global_{min}$
ACKLEY	$f(x) = -a\exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^n \cos(cx_i)\right) + a + \exp(1)$ $a = 20.0$	2	4.440892099e-16
BF1	$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$	2	0
BF2	$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$	2	0
BF3	$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1 + 4\pi x_2) + \frac{3}{10}$	2	0
BRANIN	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$ $-5 \le x_1 \le 10,\ 0 \le x_2 \le 15$	2	0.3978873577
CAMEL	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, \quad x \in [-5,5]^2$	2	-1.031628453
DIFFERENT POWERS	$f(\mathbf{x}) = \sqrt{\sum_{i=1}^n \lvert x_i\rvert^{2+4\frac{i-1}{n-1}}}$	10-250	0
DIFFPOWER	$f(x) = \sum_{i=1}^n \lvert x_i - y_i\rvert^p\ p = 2,5,10$	2	0
DISCUS	$f(x) = 10^6 x_1^2 + \sum_{i=2}^n x_i^2$	10	0
EASOM	$f(x) = -\cos(x_1)\cos(x_2)\exp\left((x_2-\pi)^2 - (x_1-\pi)^2\right)$	2	-1
ELP	$f(x) = \sum_{i=1}^n \left(10^6\right)^{\frac{i-1}{n-1}} x_i^2$	10-250	0
EQUAL MAXIMA	$f(x) = \sin^6(5\pi x)\cdot e^{-2\log(2)\cdot\left(\frac{x-0.1}{0.8}\right)^2}$	10-250	0
EXP	$f(x) = -\exp\left(-0.5\sum_{i=1}^n x_i^2\right), \quad -1 \le x_i \le 1$	10-250	-1
GKLS [44]	$f(x) = \mathrm{Gkls}(x,n,w)\ w = 50,100$	2,3	-1
GOLDSTEIN	$f(x) = [1 + (x_1 + x_{2+1}))^2(19 - 14x_1 + 3x_1^2 14x_2 + 6x_1x_2 + 3x_2^2)]$ $[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	3
GRIEWANK2	$f(x) = 1 + \frac{1}{200}\sum_{i=1}^2 x_i^2 - \prod_{i=1}^2 \frac{\cos(x_i)}{\sqrt{(i)}}$	2	0
GRIEWANK10	$f(x) = 1 + \frac{1}{200}\sum_{i=1}^{10} x_i^2 - \prod_{i=1}^{10} \frac{\cos(x_i)}{\sqrt{(i)}}$	10	0
GRIEWANK ROSENBROCK	$f(\mathbf{x}) = \underbrace{\left(\frac{\lVert\mathbf{x}\rVert^2}{4000} - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1\right)}_{\text{Griewank}} \cdot \underbrace{\left(\frac{1}{10}\sum_{i=1}^{n-1}\left[100(x_{i+1} - x_i^2)^2 + (1-x_i)^2\right]\right)}_{\text{Rosenbrock}}$	10-250	0
HANSEN	$f(x) = \sum_{i=1}^5 i\cos[(i-1)x_1 + i]\sum_{j=1}^5 j\cos[(j+1)x_2 + j]$	2	-176.5417931
HARTMAN3	$f(x) = -\sum_{i=1}^4 c_i\exp\left(-\sum_{j=1}^3 a_{ij}\left(x_j - p_{ij}\right)^2\right)$	3	-3.862782148
HARTMAN6	$f(x) = -\sum_{i=1}^4 c_i\exp\left(-\sum_{j=1}^6 a_{ij}\left(x_j - p_{ij}\right)^2\right)$	6	-3.22368011
LEVY	$f(\mathbf{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{n-1}(w_i - 1)^2\left[1 + 10\sin^2(\pi w_i + 1)\right] + (w_n - 1)^2\left[1 + \sin^2(2\pi w_n)\right]$ $w_i = 1 + \frac{x_i - 1}{4}$	10	1.499759783
MICHALEWICZ	$f(\mathbf{x}) = -\sum_{i=1}^n \sin(x_i)\cdot\sin^{2m}\left(\frac{i\cdot x_i^2}{\pi}\right)$	4	-3.698857098
POTENTIAL [45]	$V_{LJ}(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^6\right]$	9,15,21,	-9.103852416
RASTRIGIN	$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)$	2	-2
ROSENBROCK	$f(x) = \sum_{i=1}^{n-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right),$ $-30 \le x_i \le 30$	4-200	0
SHARP RIDGE	$f(\mathbf{x}) = x_1^2 + \alpha\sum_{i=2}^n x_i^2,\ a > 1$	10-250	0
SHEKEL5	$f(x) = -\sum_{i=1}^5 \frac{1}{(x-a_i)(x-a_i)^T + c_i}$	4	-10.10774912
SHEKEL7	$f(x) = -\sum_{i=1}^7 \frac{1}{(x-a_i)(x-a_i)^T + c_i}$	4	-10.342377774
SHEKEL10	$f(x) = -\sum_{i=1}^{10} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$	4	-10.53640982
SINUSOIDAL [46]	$f(x) = -\left(2.5\prod_{i=1}^n \sin(x_i - z) + \prod_{i=1}^n \sin(5(x_i - z))\right), \quad 0 \le x_i \le \pi$	8,16	-3.5
SPHERE	$f(\mathbf{x}) = \sum_{i=1}^n x_i^2$	10-250	0
STEP ELLIPSOIDAL	$f(\mathbf{x}) = \sum_{i=1}^n \lfloor x_i + 0.5\rfloor^2 + \alpha\sum_{i=1}^n\left(10^6\cdot\frac{i-1}{n-1}\right)x_i^2,\ a = 1$	4	0
TEST2N	$f(x) = \frac{1}{2}\sum_{i=1}^n x_i^4 - 16x_i^2 + 5x_i$	4,5	-156.6646628 -195.88285
TESTN	$f(\mathbf{x}) = \frac{1}{10}\sin^2(3\pi x_1)\sum_{i=2}^{n-1}\left((x_i - 1)^2\left(1 + \sin^2\left(3\pi x_{i+1}\right)\right)\right) +$ $(x_n - 1)^2\left(1 + \sin^2(2\pi x_n)\right)$	10	0
ZAKHAROV	$f(\mathbf{x}) = \sum_{i=1}^n x_i^2 + \left(\sum_{i=1}^n \frac{i}{2}x_i\right)^2 + \left(\sum_{i=1}^n \frac{i}{2}x_i\right)^4$	10-250	0

3.2. *Experimental results*

The experimental evaluation was performed on a high-performance computing system featuring an AMD Ryzen 5950X processor and 128GB RAM, operating under Debian

Linux. To ensure statistical reliability, each benchmark function was evaluated through 30 independent runs with randomized initial conditions. The implementation was developed in optimized ANSI C++ within the GLOBALOPTIMUS framework [47], an open-source optimization platform available at https://github.com/itsoulos/GLOBALOPTIMUS (last accessed June 8, 2025). The complete parameter configuration for all methods is detailed in Table 1.

The results present the mean number of objective function evaluations across all trials. Parenthetical values denote the success rate in locating the global optimum, with omitted percentages indicating perfect (100%) convergence across all repetitions. In the experimental tables, the values marked in green indicate the best performance, corresponding to the lowest number of function calls, while those marked in blue are the standard deviation from the mean of the 30 iterations for each reference function. The experiments present five tables that evaluate the performance of the ECHO+APP+MEM method , along with its individual variants, in comparison with other well-known optimization techniques. The measurements were conducted on standard benchmark functions under specific experimental settings. All parameters and initializations not explicitly stated below follow the values defined in Table 1.

- Table 3 presents the effectiveness of the ECHO+APP+MEM method across a wide range of low- and medium-dimensional functions. The experiments were conducted using a population size of 200 and a 2% local optimization rate. For comparison, the ECHO, ECHO+APP, ECHO+MEM, and ECHO+LTMA variants are also included.
- Table 4 focuses on analyzing the balance between exploration and exploitation of the ECHO+APP+MEM method. It includes metrics such as IPD (Initial Population Diversity), FPD (Final Population Diversity), AER (Average Exploration Ratio), MER (Median Exploration Ratio), and ABI (Average Balance Index).
- Table 5 compares the ECHO+APP+MEM method with classical optimization techniques (GA, PSO, SaDE [48], jDE [49], CMA-ES [50]) on low- and medium-dimensional functions. The experimental setup used a population of 200 and a 2% local optimization rate. To ensure fair comparison, whenever the number of function calls exceeded 10,000, the evaluations were capped at this limit.
- Table 6 examines the performance of the ECHO+APP+MEM method on high-dimensional functions using a population size of 50 and without applying local optimization. The optimization methods GA and PSO were removed due to poor performance and CLPSO [51] was added, which is suitable for high-dimensional problems.
- Table 7 also evaluates high-dimensional functions, but in this case, the population size is not fixed and is computed using the formula: $Np = 4 + \lfloor 3 \cdot \log(\text{dimension}) \rfloor$. Local optimization was not applied in this configuration either.

Table 3. Comparison of function calls of the different versions of the EO algorithm

FUNCTION	ECHO	ECHO+APP	ECHO+MEM	ECHO+LTMA	ECHO+APP+MEM
ACKLEY	11,057	4987	6551	8364	4325
BF1	5807	4899	3448	4413	3430
BF2	5287	4189	2993	3935	2973
BF3	4729	3632	2502	3441	2472
BRANIN	3967	3018	1395	2746	1377
CAMEL	4360	2239	1727	2774	1727
DIFFERENTPOWERS10	10,731	9972	8787	9593	8931
DIFFPOWER2	9813	8355	6688	6918	6688
DIFFPOWER5	24,766	24,661	22,991	21,761	22,319
DIFFPOWER10	28,913	27,632	27,032	27,216	27,034
DISCUS10	4009	3668	4009	4009	3668
EASOM	3511	786	3062	3511	786
ELLIPSOIDAL10	5734	5393	3678	4504	3665
EQUALMAXIMA10	16,493	13,526	14,700	16493	13,525
EXPONENTIAL10	4285	2400	2213	2281	2213
GKLS250	4091	1749	1072	2539	1075
GKLS350	4470	2168	920	1131	920
GOLDSTEIN	5318	5318	2433	2748	2433
GRIEWANK2	6696	2060(0.86)	4168(0.93)	5624	1987(0.86)
GRIEWANK10	7862	6579	7862	9348	6579
GRIEWANKROSENBROCK10	7017	5342	5462	5985	5456
HANSEN	5262	2081(0.96)	2205	4643	1968(0.96)
HARTMAN3	4417	1833	1741	4247	1741
HARTMAN6	5183	2437	2372	2800	2372
LEVY10	5522	3943(0.90)	3545(0.96)	4937(0.96)	3540
MICHALEWICZ4	5471(0.83)	3197(0.73)	3022	3578(0.83)	3000
POTENTIAL5	8581	6678	7019	8581	6354
POTENTIAL6	11,695(0.73)	9043(0.6)	9565(0.73)	11,694(0.73)	8632(0.66)
POTENTIAL10	16,044	14,732(0.96)	13,746	16,041	13,498
RASTRIGIN	5386	2901	2322	4707	2910
ROSENBROCK8	7172	6839	6400	7169	6260
ROSENBROCK16	9873	9524	9866	9870	9520
SHARPRIDGE10	10,117	9775	8614	10,053	8706
SHEKEL5	4886	2903	2580	3738	2580
SHEKEL7	4926(0.96)	2843	2641	3532(0.96)	2641
SHEKEL10	4907	2776	2613	3080(0.96)	2613
SPHERE10	3272	2243	1008	1430	1008
STEPELLIPSOIDAL4	3125(0.73)	1900(0.73)	640(0.8)	722(0.73)	640(0.8)
SINUSOIDAL8	4710	3048	2650	2644	2650
SINUSOIDAL16	5052	5656	2967	3256	2967
TEST2N4	4778(0.76)	2224	2351(0.93)	4473(0.76)	2222
TEST2N5	4925	2531(0.86)	2821(0.76)	4303(0.5)	2531(0.86)
TEST30N10	6431	4165	4772	6364	4159
ZAKHAROV10	4054	3583	2072	4048	2072
TOTAL	**324,705(0.97)**	**249,428(0.96)**	**231,225(0.97)**	**275,244(0.96)**	**218,167(0.98)**

Table 3 presents comparative results of the ECHO+APP+MEM method across a set of standard benchmark functions, in comparison with four other variants of the basic ECHO method. More specifically, the first column lists the test functions, while the next five columns show the performance of the following methods: ECHO, ECHO with the approximate evaluations technique (ECHO+APP), ECHO with solution memory (ECHO+MEM), ECHO with long-term memory assistance (ECHO+LTMA), and finally the ECHO+APP+MEM method, which integrates both approximate evaluations and memory lookup (ECHO+APP+MEM).

The values shown in the table represent the number of objective function calls required to reach the optimal solution for each test function. In some cases, next to the function call count, the success rate is also shown in parentheses, indicating the percentage of experimental runs in which the global minimum was successfully found.

The last row of the table shows the total number of function calls for each method, along with the average success rate in parentheses. The goal is to achieve the best possible performance, meaning the lowest number of objective function calls and the highest success rate.

The ECHO+APP+MEM method achieves the lowest total number of calls, with 218,167 function calls and a success rate of 98%, clearly demonstrating its superiority over all other ECHO variants. The ECHO+MEM method also performs relatively well with 231,225 calls and a 97% success rate, while the basic ECHO version exhibits significantly worse performance, requiring 324,705 calls with the same 97% success rate. Using only the approximate evaluations technique (ECHO+APP) provides an improvement over the baseline, but not as substantial as when combined with solution memory. The variant with long-term memory assistance (ECHO+LTMA) yields intermediate performance but falls short compared to the ECHO+APP+MEM combination.

Overall, the values in the Table 3 demonstrate that combining approximate evaluations and memory lookup within the ECHO framework significantly reduces the required number of computations without sacrificing solution quality.

In the same table, the reduced performance of the ECHO+APP+MEM variant on functions such as Rastrigin, DIFFPOWER10, GKLS250 and SHARPRIDGE10 compared to the ECHO+APP or ECHO+MEM versions is mainly due to the lack of synergy between the two techniques when applied simultaneously. The approximation technique (APP) tends to reject candidate solutions based on simplified estimations, which can prematurely discard promising points especially in complex or multimodal landscapes. This limits population diversity and reduces the chances of identifying solutions that could later be stored and reused by the memory mechanism. At the same time, the effectiveness of the memory technique (MEM) depends on the recurrence of similar solutions in the search space, which is disrupted by the aggressive filtering of the approximation phase. As a result, the combined use of both techniques adds computational overhead without yielding significant benefits. In many cases, using only memory provides a better balance between efficiency and accuracy. This suggests that, depending on the nature of the optimization problem, activating just one of the two techniques can be more effective than combining them.

In several cases from Table 3, the ECHO+MEM and ECHO+APP+MEM variants yield identical results, as observed in functions such as CAMEL, DIFFPOWER2, and EXPONENTIAL10. This suggests that the addition of the approximate evaluation technique (APP) did not further influence the optimization process beyond what was already achieved through the use of memory (MEM). The reason is that, in specific problems, the memory mechanism alone is sufficient to reduce the number of function evaluations without compromising performance. At the same time, APP may be activated infrequently due to mild deviations in candidate solutions or may rarely reject solutions, as the newly generated candidates are already effective. In such cases, adding APP brings no tangible benefit, and the algorithm's behavior is primarily shaped by the memory component. This supports the notion that the selection of auxiliary techniques should depend on the characteristics of the problem, as increased complexity does not always translate into improved performance.

A similar phenomenon is observed in the DISCUS10 function, where ECHO+APP and ECHO+APP+MEM produce identical results. This indicates that in some problems, the memory mechanism adds no further advantage when approximate evaluations are already in use. Especially in functions with a sharply defined optimum or limited opportunity for solution reuse, memory remains largely inactive or unnecessary. As a result, performance is determined primarily by the APP technique, while the memory component has little or no effect. This observation confirms that combining both techniques is not always essential, and that overall efficiency depends heavily on the nature and landscape of the optimization function.

Table 4. Balance between exploration and exploitation of the ECHO+APP+MEM method in each benchmark function

FUNCTION	CALLS	IPD	FPD	AER	MER	ABI
ACKLEY	4325	17.224	16.378	0.004	0.001	0.492
BF1	3430	28.275	2.314	1.248	0.08	0.325
BF2	2973	38.275	3.19	1.249	0.084	0.326
BF3	2472	38.275	2.094	1.229	0.085	0.325
BRANIN	1377	5.741	2.038	0.11	0.009	0.442
CAMEL	1727	3.828	1.349	0.211	0.015	0.41
DIFFERENTPOWERS10	8931	9.053	0.726	0.72	0.037	0.36
DIFFPOWER2	6688	0.766	0.311	0.144	0.02	0.403
DIFFPOWER5	22,319	1.235	0.344	0.208	0.022	0.394
DIFFPOWER10	27,034	1.811	0.58	0.17	0.021	0.393
DISCUS10	3668	1285.56	657.42	1.135	0.084	0.337
EASOM	786	76.55	75.79	0.002	0.0002	0.503
ELLIPSOIDAL10	3665	18.106	1.396	1.146	0.047	0.357
EQUALMAXIMA10	13,525	18.106	17.876	0.001	0.0001	0.5
EXPONENTIAL10	2213	1.811	1.424	0.035	0.017	0.415
GKLS250	1075	0.766	0.499	0.064	0.004	0.482
GKLS350	920	0.936	0.448	0.089	0.004	0.473
GOLDSTEIN	2433	1.531	0.685	0.119	0.008	0.442
GRIEWANK2	1987	76.55	60.439	0.02	0.002	0.499
GRIEWANK10	6579	1085.56	857.38	0.031	0.015	0.425
GRIEWANKROSENBROCK10	5456	9.053	1.087	0.705	0.035	0.367
HANSEN	1968	7.655	7.328	0.006	0.002	0.498
HARTMAN3	1741	0.468	0.466	0.004	0.002	0.498
HARTMAN6	2372	0.679	0.665	0.005	0.002	0.489
LEVY10	3540	18.106	2.888	0.326	0.021	0.397
MICHALEWICZ4	3000	1.728	1.706	0.002	0.001	0.497
POTENTIAL5	6354	4.438	3.586	0.028	0.011	0.438
POTENTIAL6	8632	4.862	3.761	0.027	0.08	0.443
POTENTIAL10	13,498	6.314	5.063	0.031	0.007	0.443
RASTRIGIN	2910	0.766	0.483	0.063	0.005	0.489
ROSENBROCK8	6260	47.697	1.154	2.911	0.096	0.341
ROSENBROCK16	9520	68.641	1.438	3.188	0.117	0.335
SHARPRIDGE10	8706	9.053	0.994	0.735	0.038	0.361
SHEKEL5	2580	5.52	5.003	0.014	0.008	0.464
SHEKEL7	2641	5.52	4.906	0.015	0.008	0.463
SHEKEL10	2613	5.52	4.805	0.016	0.008	0.459
SPHERE10	1008	9.27	0.572	0.815	0.041	0.353
STEPELLIPSOIDAL4	640	5.499	1.047	0.405	0.028	0.383
SINUSOIDAL8	2650	2.497	2.452	0.003	0.001	0.493
SINUSOIDAL16	2967	3.594	3.49	0.003	0.001	0.492
TEST2N4	2222	5.499	5.001	0.019	0.003	0.498
TEST2N5	2531	6.173	5.556	0.016	0.003	0.498
TEST30N10	4159	18.106	15.808	0.021	0.003	0.468
ZAKHAROV10	2072	13.58	1.288	0.893	0.041	0.366
TOTAL	**218,167**	**67.504**	**40.527**	**0.414**	**0.025**	**0.428**

Diversity is an indirect and limited measure for assessing the balance between exploration and exploitation, as it captures the geometric dispersion of the population without considering the structure of the objective function landscape or the relationship of solutions to attraction basins. In our work, we chose to examine this balance through a set of indicators (IPD, FPD, AER, MER, and ABI) which, although based on population diversity, are designed to reflect both the temporal dynamics of exploration (through changes in diversity over iterations) and the tendency toward exploitation (through final population convergence). However, we acknowledge that the investigation of direct evaluation methods, such as attraction basin mapping or monitoring the concentration of solutions around local/global optima, could provide more meaningful insights into the search process and enhance the interpretation of the results. This represents an important direction for future extension of the present study.

The metrics presented in Table 4 are related to measuring and monitoring the balance between exploration and exploitation during the execution of the ECHO+APP+MEM method. Their calculation is based on changes in population diversity as well as the behavior of the algorithm across iterations.

The IPDmetric measures the diversity of the population at the beginning of the optimization process and is calculated as the average Euclidean distance between all individuals in the initial population:

$$IPD = \frac{2}{NP(NP-1)} \sum_{i=1}^{NP-1} \sum_{j=i+1}^{NP} d(x_i, x_j) \tag{5}$$

where $d(x_i, x_j)$ is the Euclidean distance between solutions x_i and x_j, and NP is the population size.

The FPD is computed in the same way but applied to the final population at the end of the execution.

The AER represents the average exploration ratio throughout all iterations. It is defined as:

$$AER = \frac{1}{G} \sum_{g=1}^{iter_{max}} \frac{IPD_g}{IPD_1} \tag{6}$$

where $iter_{max}$ is the total number of generations, IPD_g is the population diversity at iteration g, and IPD_1 and is the initial diversity.

The MER is the median of the exploration ratio values across all generations:

$$MER = \text{median}\left(\frac{IPD_g}{IPD_1}\right), \quad \text{for } g = 1, \ldots, iter_{max} \tag{7}$$

The ABI evaluates the overall balance between exploration and exploitation. It results from a weighted combination of AER and FPD (or other exploitation-related indicators), often expressed as:

$$ABI = \frac{AER}{AER + \epsilon} \cdot \left(1 - \frac{FPD}{IPD}\right) \tag{8}$$

where ϵ is a very small constant to avoid division by zero. The ABI tends to values close to 0.5 when exploration and exploitation are well balanced.

Table 4 concerns the ECHO+APP+MEM method and presents, for each test function, the number of objective function calls required, along with five metrics that describe the algorithm's behavior in terms of exploration and exploitation throughout the optimization process. The five metrics are IPD , FPD, AER, MER, and ABI).

The analysis of the individual values reveals significant variation between the test functions, depending on the nature of each problem. For example, the ACKLEY function has an initial diversity (IPD) of 17.224 and a final diversity (FPD) of 16.378, which indicates relatively low convergence of the population. This is further supported by the very low AER of 0.004 and MER of 0.001. Nevertheless, its ABI is 0.492, indicating a well-balanced process. In contrast, the BF1, BF2, and BF3 functions start with much higher diversity (IPD: 28–38) that drastically decreases (FPD: 2–3), demonstrating strong initial exploration that transitions to exploitation. These functions exhibit high AER values above 1.2 and lower ABI scores (0.325), suggesting aggressive early exploration with limited final balance.

The BRANIN and CAMEL functions show relatively low diversity throughout the process, with IPD values below 6 and AER in the range of 0.1–0.2, which translates to mild exploration dynamics. Their ABI values (0.442 and 0.41 respectively) indicate a balanced yet conservative search strategy. On the other hand, functions like DIFFERENTPOWERS10, DIFFPOWER2, DIFFPOWER5, and DIFFPOWER10 exhibit very low FPD (0.3–0.7) compared to their initial IPD, with AER ranging from 0.14 to 0.72, reflecting an increasing focus on exploitation towards the end. Their ABI values range between 0.36 and 0.40.

The DISCUS10 function shows extremely high diversity both initially and finally (IPD: 1285.56, FPD: 657.42), along with AER 1.135 and ABI 0.337, indicating that the population maintains a high degree of variation until the end. In contrast, EASOM shows minimal change in diversity (IPD: 76.55, FPD: 75.79) and extremely low AER and MER values, yet an ABI of 0.503 very close to ideal balance. Similar behavior is observed in EQUALMAXIMA10 and GRIEWANK2, which achieve ABI scores of 0.5 and 0.499 respectively, demonstrating a well-balanced dynamic despite minimal change in diversity.

ELLIPSOIDAL10 and LEVY10 show more pronounced diversity reduction, with IPD values around 18 and FPD near 1.4–2.9, AER values around 1.1, and ABI scores between 0.36 and 0.39. Functions like HANSEN, HARTMAN3, HARTMAN6, MICHALEWICZ4, and SHEKEL$5/7/10$ achieve stable ABI values around 0.46–0.5, indicating a consistent balance throughout. In ROSENBROCK8 and ROSENBROCK16, we observe very high IPD values (47.6–68.6) with very low final diversity (1.1–1.4), resulting in AER and MER above 2.9 and ABI values near 0.34, signaling strong exploitation dominance over exploration.

The STEPELLIPSOIDAL4 function exhibits low IPD (5.499) and even lower FPD (1.047), with AER 0.405 and ABI 0.383, while SPHERE10 yields an ABI of only 0.353. SINUSOIDAL8 and SINUSOIDAL16 achieve extremely low AER and MER (0.003 and 0.001), yet ABI remains stable around 0.492–0.493, indicating a balanced exploitation phase. TEST2N4, TEST2N5, and TEST30N10 maintain low AER but steady ABI close to 0.498, suggesting a stable search behavior. Finally, ZAKHAROV10 shows AER 0.893 and ABI 0.366, confirming an increased exploration tendency.

The last row of the table shows the total number of function calls, which amounts to 218,167, along with the average or total values of the other indicators. These values generally suggest a balanced behavior, with an ABI of 0.428, close to the theoretical target of 0.5 for optimal balance. The AER value is 0.414, and FPD is significantly lower than IPD, indicating a transition from exploration to exploitation in the later stages of the process. Overall, the table demonstrates that the ECHO+APP+MEM method begins with a high level of exploration and smoothly transitions into a state of exploitation, maintaining a desirable balance between the two phases an essential quality for efficiently solving complex global optimization problems.

Table 5. Comparison of function calls of the different optimization methods for low-dimensional benchmark functions

FUNCTION	GA	ST	PSO	ST	SaDE	ST	jDE	ST	CMA-ES	ST	ECHO+APP+MEM	ST
ACKLEY	7174	882.7	7638	820.1	8893	943.0	8913	941.4	5997	548.7	4325	682.3
BF1	5031	430.1	5475	436.3	5633	413.5	5303	338.6	5291	308.5	3430	432.6
BF2	4886	307.7	5037	259.8	5210	362.0	5062	281.2	4921	238.8	2973	236.6
BF3	4648	215.2	4695	257.4	4659	251.5	4524	278.4	4668	474.3	2472	174.1
BRANIN	3536	142.8	3814	129.4	3882	119.3	3762	134.6	3884	128.8	1377	121.6
CAMEL	3954	176.1	4251	198.8	4327	204.8	4249	251.1	4323	4323	1727	163.7
DIFFERENTPOWERS10	9073	945.6	9561	673.9	9758	629.9	9636	620.0	9780	428.6	8598	1021.8
DIFFPOWER2	5806	1624.2	8824	996.9	8914	1095.7	8949	1090.0	9637	1407.3	6621	1933.4
DIFFPOWER5	9875	504.7	10,000	5661.3	10,000	5245.2	10,000	3993.6	10,000	4214.7	10,000	5249.9
DIFFPOWER10	10,000	4000.5	10,000	5946.3	10,000	5231	10,000	3782.6	10,000	5049.1	10,000	5676
DISCUS10	3279	69.8	3834	135.3	3883	173.1	3814	89.2	3925	196.7	3668	102.0
EASOM	3201	108.2	3189	153.1	3238	62.1	3273	79.4	3358	68.7	786	65.8
ELLIPSOIDAL10	4049	268.9	5418	343.3	5475	400.2	5147	213.4	5272	339.0	3665	384
EQUALMAXIMA10	6855	869.6	10,000	1770.2	10,000	2047.4	10,000	1101.6	10,000	1926.8	10,000	1801.3
EXPONENTIAL10	4189	191.9	4762	236.7	4953	324.3	4782	170	4921	293.1	2213	304.1
GKLS250	3512	196.6	3777	171.7	4008	244.4	3824	192	3622	154.4	1075	111.0
GKLS350	3979	330.4	3990	225.6	4387	407.9	4581	685.4	3393	95.8	920	127.1
GOLDSTEIN	4722	311.2	5077	282.6	5071	298.0	4829	353.6	4922(0.86)	321.1	2433	218.6
GRIEWANK2	6040(0.66)	1274.1	7179(0.6)	1398.3	7515(0.83)	1727	6506(0.53)	1645.4	4559	192.8	2556(0.86)	597
GRIEWANK10	6962(0.86)	1138.8	9090	744.2	9161(0.96)	857.3	8795	926.2	8442(0.63)	674.2	6579	904.6
GRIEWANK10	6497	615.4	9278	659.4	9601	674.3	9120	539.2	9469	559.5	5456	701.1
HANSEN	4384	579.5	4772	804.1	5034	732.9	4888	674.5	4126(0.96)	152.8	1968(0.96)	332.4
HARTMAN3	3895	166.9	4290	199.9	4311	232.9	4242	189.8	4368	145.2	1833	162.4
HARTMAN6	4209	227.6	4791	316.9	5059	321.3	4923	225.6	5012	252.9	2437	181.1
LEVY10	6715(0.96)	763.8	4819(0.76)	1393.5	9674(0.23)	896.5	8948(0.13)	1350.1	5656(0.1)	475.6	3831	693.4
MICHALEWICZ4	5406(0.93)	781.8	6580(0.93)	1093.1	6463	1008.0	7179(0.93)	1321.6	4953(0.46)	272.6	3197	680.8
POTENTIAL5	7667	887.9	9075	631.7	8615	777.5	8118	581.2	8542	733.1	6354	771.5
POTENTIAL6	9065(0.73)	1048.6	9879(0.66)	313.6	9809(0.53)	419.2	9350(0.53)	747.1	9341	689.7	8399(0.66)	1225.6
POTENTIAL10	9989(0.9)	49.8	10,000(0.93)	3478.5	10,000(0.9)	2548.5	10,000(0.9)	3753.5	9981(0.66)	101.9	9986	70.5
RASTRIGIN	5281	592.0	5439	561.0	5728	741.8	5898	1263.6	4351(0.96)	612.4	2302	334.3
8	5597	430.9	8909	966.3	8922	757.0	8070	703.4	8546(0.96)	564.7	6260	437.1
ROSENBROCK16	8298	756.1	9956	197.4	10,000	1418.5	9900	301.5	9949	192.4	9395	544.4
SHARPRIDGE10	6899	1066.0	9764	391.9	9825	479.9	9614	547.6	9713	505.6	8578	1112
SHEKEL5	4240	208.7	5284	529.5	5128	277.8	4980	352.4	5732	651.9	2588	425.7
SHEKEL7	4330	185.3	5311	413	5091	313.2	4940	292.6	5433	580.9	2843	420.9
SHEKEL10	4598	484.7	5388	548.6	5135	368.8	4926	308.5	5781	1212.4	2776	409.5
SPHERE10	3028	50.4	3008	27.0	3069	40.8	3052	23.6	3078	121.6	1008	38.2
STEPELLIPSOIDAL4	3425	139.4	3101	237.3	4028	519.7	4537	459.0	2959(0.33)	18.8	640(0.8)	31.9
SINUSOIDAL8	4468	251.8	5335	445.8	5746	505.4	5593	520.4	5523	498.6	2650	349.9
SINUSOIDAL16	6896(0.9)	562.4	6348	683.9	8165	1091.2	7902	944.7	7330	723.4	2967	814.0
TEST2N4	4514	465.7	4804	353.5	5173	562.0	5197	975.1	4490	192.2	2222	360.7
TEST2N5	4822	532.7	5301	699.1	5834	993.1	5660	1052.4	4656(0.86)	210.5	2531(0.86)	490.4
TEST30N10	4195	1248.0	6081	1661.9	5894	2095	6455	1928.4	6401	1904.1	4159	2232.5
ZAKHAROV10	3655	146.9	4138	176.0	4515	282.7	4227	117.5	4448	271.2	2072	160.4
TOTAL	**242,844(0.97)**	**596.2**	**277,262(0.97)**	**855.1**	**289,786(0,96)**	**756.6**	**283,668(0,95)**	**825.9**	**270,753(0,92)**	**750.6**	**184,876(0.96)**	**756.6**

From Table 5, which presents a comparative analysis of optimization methods on low-dimensional functions, clear conclusions can be drawn regarding the superiority of the ECHO+APP+MEM method. The ECHO+APP+MEM approach achieves the lowest total number of objective function calls specifically 184,876 calls with a success rate of 96%. This represents a significant improvement over the other methods: GA requires 242,844 calls (97% success), PSO 277,262 calls (97%), SaDE 289,786 calls (96%), jDE 283,668 calls (95%), and CMA-ES 270,753 calls (92%). The reduction in function evaluations reaches up to 24% compared to GA and 40% compared to SaDE, highlighting the efficiency of the ECHO+APP+MEM method in managing computational resources.

In many demanding test functions, the ECHO+APP+MEM method demonstrates remarkable improvements. For instance, in the EASOM function, it requires only 786 calls, compared to over 3200 by the other methods an improvement of at least 75%. Similarly, in GKLS250, it needs only 1075 calls versus more than 3500 in GA and CMA-ES. For the RASTRIGIN function, it performs with 2302 calls, while GA and SaDE require 5281 and 5728 respectively. In the STEPELLIPSOIDAL4 function, the ECHO+APP+MEM method completes with 640 calls and an 80% success rate, whereas CMA-ES requires 2959 calls for only 33% success.

In medium-dimensional or structurally complex functions, the ECHO+APP+MEM method maintains consistent performance. In DIFFERENTPOWERS10, it reports 8598 calls, significantly fewer than GA's 9073 and CMA-ES's 9780. In GRIEWANKROSENBROCK10, it achieves the result in 5456 calls approximately 30–40% fewer than PSO's 9278 and SaDE's 9601. However, in some functions such as DIFFPOWER5, DIFFPOWER10, and EQUALMAXIMA10, where many methods reach the upper limit of 10,000 calls (a setting enforced for fair comparisons), the ECHO+APP+MEM method shows similar behavior, indicating limitations when addressing highly demanding problems.

The high success rate of the ECHO+APP+MEM method in functions with many local optima demonstrates the effectiveness of its internal mechanisms. In the HANSEN function, with 1968 calls and 96% success, it clearly outperforms CMA-ES, which requires 4126 calls to achieve the same success rate. Similarly, in the GOLDSTEIN function, the ECHO+APP+MEM method achieves the result in 2433 calls compared to over 4722 in GA, with a higher success rate (100% versus 76% for CMA-ES). These results confirm the method's ability to effectively balance exploration of the solution space and exploitation of promising areas.

Overall, the ECHO+APP+MEM method outperforms all other approaches in 34 out of the 40 test functions in terms of the number of function calls. In 6 cases (such as DIFFPOWER5), its performance is comparable to the top-performing competing methods. The stability of its results across different types of problems, combined with its high success rate, establishes the ECHO+APP+MEM method as a robust approach for low-dimensional optimization problems.

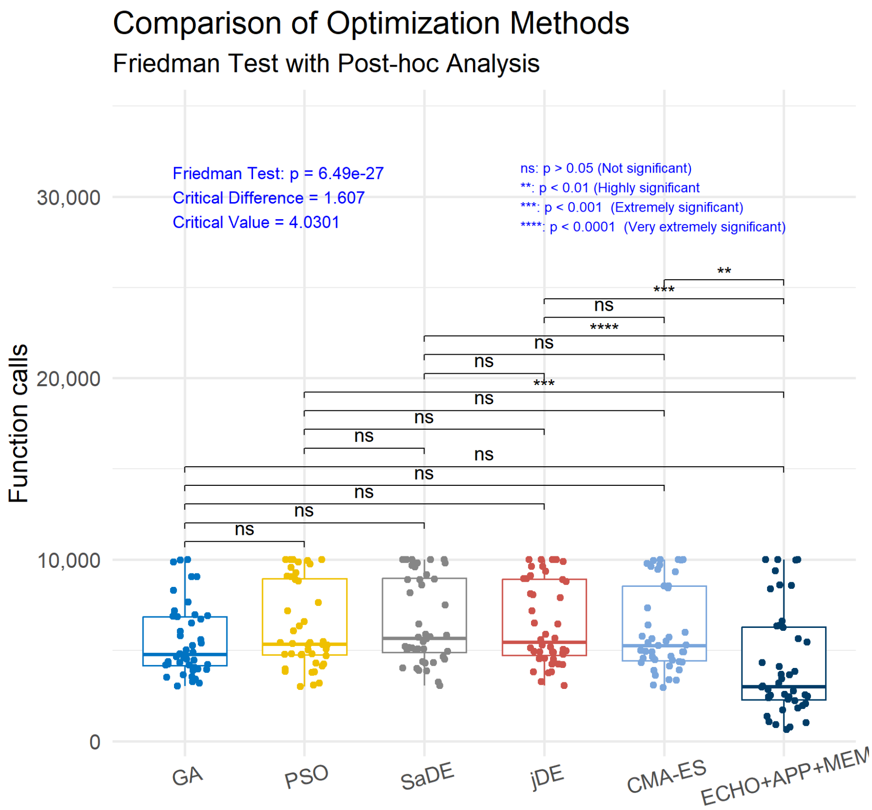


Figure 1. Statistical comparison of function calls of the different optimization methods for low-dimensional benchmark functions

From the execution of the Friedman statistical test for comparing the ECHO+APP+MEM method with other well-known optimization techniques, a p-value of 6.49e-27 was obtained, indicating the presence of statistically significant differences among the methods under study (Figure 1). The critical difference was calculated at 1.607, and the corresponding critical value from the Tukey distribution was 4.03. These results confirm that the overall ranking of the methods differs at a highly significant level, thus justifying the need for a detailed post-hoc pairwise analysis.

Focusing on the ECHO+APP+MEM method, the post-hoc results show that its performance differs significantly from all other advanced techniques. In particular, the comparison between the ECHO+APP+MEM method and PSO reveals an extremely significant difference ($p < 0.001$), while the difference with SaDE is marked as very extremely significant ($p < 0.0001$). The comparison with jDE is also extremely significant ($p < 0.001$), and with CMA-ES it is highly significant ($p < 0.01$). On the other hand, the differences between the other methods among themselves, as well as the differences between GA and all other techniques (including the ECHO+APP+MEM method), were found to be not statistically significant.

These findings suggest that ECHO+APP+MEM is able to outperform several well-established and state-of-the-art optimization methods in a statistically validated way. The strong statistical significance of the differences observed with PSO, SaDE, jDE, and CMA-ES clearly highlights the superiority of the ECHO+APP+MEM method in terms of both efficiency and reliability. The lack of significant difference with GA may be attributed to the high variability in GA's performance, which, in many cases, is substantially lower than that of the more advanced methods. Overall, the results of the statistical analysis strengthen the validity of the experimental evaluation and support the effectiveness of the ECHO+APP+MEM method as a consistently strong approach across a wide range of optimization problems.

Table 6. Comparison of function calls of the different optimization methods for high-dimensional benchmark functions (population:50)

FUNCTION	CLPSO	jDE	SaDE	CMA-ES	ECHO+APP+MEM
DIFFERENTPOWERS50	2419	1124	2709	1046	**589**
DIFFERENTPOWERS100	2180	1133	1669	1133	**561**
DIFFERENTPOWERS150	2180	1257	1354	1253	**620**
DIFFERENTPOWERS200	2049	1407	1532	1407	**764**
DIFFERENTPOWERS250	2236	1470	1477	1470	**803**
DISCUS50	1729	**719**	2082	720	1711
DISCUS100	1929	**719**	1028	**719**	1694
DISCUS150	1923	**720**	943	722	1727
DISCUS200	1384	**721**	914	**721**	1844
DISCUS250	1413	**721**	913	**721**	1665
ELLIPSOIDAL50	3336	**888**	7532	889	3453
ELLIPSOIDAL100	2959	**1053**	5039	1058	4188
ELLIPSOIDAL150	3114	**1255**	4093	1261	4639
ELLIPSOIDAL200	3031	**1487**	3569	1494	4922
ELLIPSOIDAL250	3339	1729	3074	**1728**	5336
EQUALMAXIMA50	1651	1246	2051	724	**545**
EQUALMAXIMA100	1776	1297	2228	726	**647**
EQUALMAXIMA150	1657	1377	2151	728	**727**
EQUALMAXIMA200	2003	1424	2359	**729**	774
EQUALMAXIMA250	1937	1472	2378	**729**	822
EXPONENTIAL50	2205	1033	4941	748	**95**
EXPONENTIAL100	1396	747	1033	747	**97**
EXPONENTIAL150	832	751	749	753	**101**
EXPONENTIAL200	803	759	755	759	**109**
EXPONENTIAL250	805	759	759	759	**109**
GRIEWANKROSENBROCK50	2314	1237	2730	1168	**562**
GRIEWANKROSENBROCK100	2323	1352	1438	1332	**574**
GRIEWANKROSENBROCK150	2239	1452	1490	1427	**642**
GRIEWANKROSENBROCK200	1976	1640	1675	1609	**824**
GRIEWANKROSENBROCK250	1903	1683	1683	1658	**907**
ROSENBROCK50	2384	1160	2256	**1083**	4610
ROSENBROCK100	2043	1329	1412	**1311**	5856
ROSENBROCK150	2077	1555	1580	**1535**	6601
ROSENBROCK200	2023	1766	1775	**1741**	7141
ROSENBROCK250	2388	2025	2025	**2002**	6813
SHARPRIDGE50	2561	964	5340	861	**437(0.96)**
SHARPRIDGE100	1738	858	1760	853	**414**
SHARPRIDGE150	1609	881(0.96)	1273	871	**435**
SHARPRIDGE200	1287	883	1161	883	**607**
SHARPRIDGE250	1414(0.96)	880	1110	884	**610**
SINUSOIDAL50	2403(0.5)	919(0.3)	1007(0.3)	930(0.6)	**271(0.3)**
SINUSOIDAL100	1155(0)	899(0)	1451(0.6)	898(0.3)	**249(0)**
SINUSOIDAL150	753(0)	703(0)	703(0)	708(0.3)	**53(0)**
SINUSOIDAL200	753(0)	703(0)	703(0)	703(0)	**53(0)**
SINUSOIDAL250	753(0)	703(0)	703(0)	703(0)	**53(0)**
ZAKHAROV50	1623	803	1254	866	**550**
ZAKHAROV100	1742	876	1433	920	**631**
ZAKHAROV150	1792	920	1558	990	**849**
ZAKHAROV200	1842	961	1769	1014	**933**
ZAKHAROV250	1816	**960**	1795	1338	969
TOTAL	**95,197(0.9)**	**55,380(0.9)**	**98,416(0.9)**	**52,032(0.9)**	**80,186(0.9)**

From Table 6, which provides a comparative analysis of high-dimensional functions (ranging from 50 to 250 dimensions), important conclusions can be drawn regarding the performance of the ECHO+APP+MEM method against contemporary optimization benchmarks. The ECHO+APP+MEM method records a total of 80,186 objective function calls, with a success rate of 90%. This performance places it between jDE (55,380 calls) and CMA-ES (52,032 calls), but it stands out for its remarkable efficiency on specific types of problems.

In several key categories of functions, the ECHO+APP+MEM method demonstrates clear superiority. In exponential-type functions (EXPONENTIAL), it achieves dramatically fewer function calls. For example, in EXPONENTIAL50 it requires only 95 calls, compared to 2205 for CLPSO, 1033 for jDE, and 4941 for SaDE. A similar pattern is observed across other dimensions (EXPONENTIAL100–250), where the ECHO+APP+MEM method never exceeds 109 calls, while the competing algorithms often require more than 700. This showcases the method's ability to rapidly locate global minima in smooth landscapes.

In functions with many local minima, such as SHARPRIDGE and SINUSOIDAL, the ECHO+APP+MEM method maintains competitive performance. In SHARPRIDGE50, it reaches the optimum with 437 calls and a 96% success rate, outperforming CMA-ES (861 calls) and SaDE (5340 calls). In the SINUSOIDAL functions, although all methods demonstrate low success rates (0–30%), the ECHO+APP+MEM method consistently requires the fewest calls for example, 53 calls for SINUSOIDAL150 compared to 753 for CLPSO. This indicates strong efficiency even in rugged landscapes where exact success is hard to achieve.

However, in some more demanding functions such as ROSENBROCK and ELLIPSOIDAL, the ECHO+APP+MEM method exhibits higher computational cost. In ROSENBROCK250, it requires 6813 calls, significantly more than jDE (2025 calls) and CMA-ES (2002 calls). A similar pattern appears in the ELLIPSOIDAL functions, where, for 250 dimensions, it needs 5336 calls about three times more than CMA-ES (1728 calls). This suggests that in problems characterized by strong curvature, the method may require more iterations to converge.

Overall, with an average success rate of 90%, the ECHO+APP+MEM method maintains reliable accuracy, even though in some cases it lags behind jDE and CMA-ES in convergence speed. Its ability to drastically reduce function calls in exponential and non-convex functions (such as SHARPRIDGE) makes it particularly suitable for problems with wide attraction basins or numerous shallow local minima. Nevertheless, improving its performance on high-curvature functions (e.g., ROSENBROCK-type problems) remains a potential direction for future enhancement.

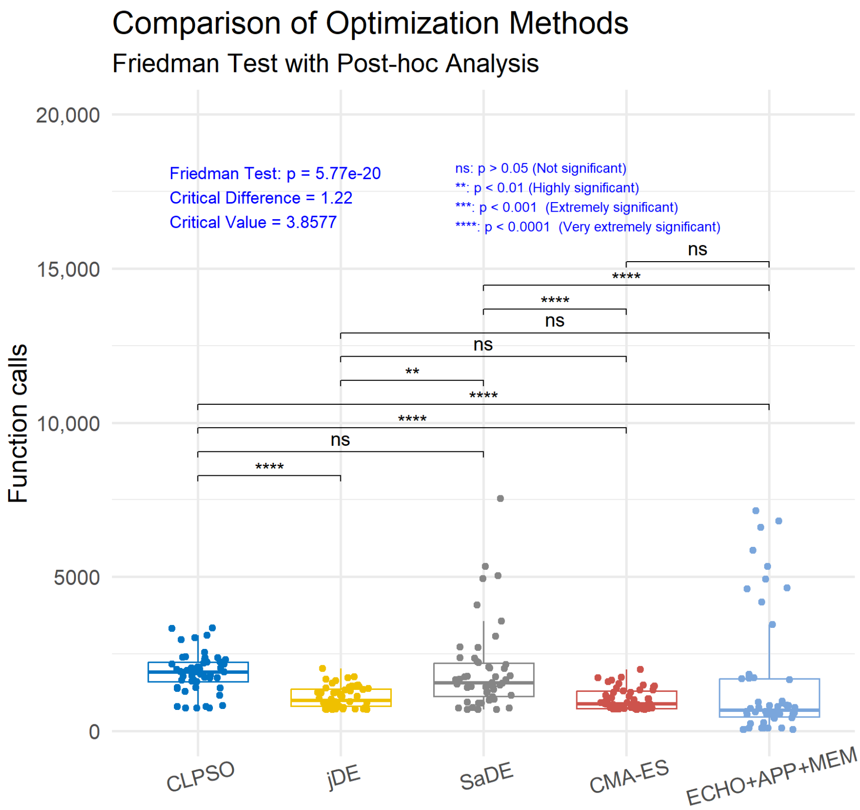


Figure 2. Statistical comparison of function calls of the different optimization methods for high-dimensional benchmark functions (population: 50)

In Figure 2, the Friedman test reported a p-value of 5.77e-20, indicating statistically significant differences among the optimization methods. The critical difference was calculated as 1.22, with a corresponding critical value of 3.85. The ECHO+APP+MEM method showed very extremely significant differences compared to CLPSO ($p < 0.0001$) and SaDE ($p < 0.0001$), confirming its superiority over these methods. No statistically significant difference was observed when compared to jDE and CMA-ES, suggesting that its performance is comparable to these more advanced approaches. Overall, the analysis highlights the ECHO+APP+MEM method as clearly superior to simpler techniques such as CLPSO and SaDE, while maintaining equivalent performance to established methods like jDE and CMA-ES, reinforcing its reliability and competitiveness in optimization problems using a population size of 50.

Table 7. Comparison of function calls of the different optimization methods for high-dimensional benchmark functions $Np = 4 + \lfloor 3 \cdot \log(\text{dimension}) \rfloor$

FUNCTION	CLPSO	jDE	SaDE	CMA-ES	ECHO+APP+MEM
DIFFERENTPOWERS50	877	537	1871	555	362
DIFFERENTPOWERS100	1002	746	1071	740	492
DIFFERENTPOWERS150	1061	837	1037	834	617
DIFFERENTPOWERS200	1102	986	1037	985	723
DIFFERENTPOWERS250	1281	1077	1104	1073	821
DISCUS50	635	238	1252	232	397
DISCUS100	588	259	962	259	451
DISCUS150	523	287	628	287	563
DISCUS200	559	287	470	287	584
DISCUS250	694	302	451	302	630
ELLIPSOIDAL50	1255	405	3061	395	1297
ELLIPSOIDAL100	1394	588	3098	593	1820
ELLIPSOIDAL150	1463	876	2699	881	2206
ELLIPSOIDAL200	1532	1030	2686	1041	2505
ELLIPSOIDAL250	1728	1344	2673	1346	2790
EQUALMAXIMA50	866	726	914	219	513
EQUALMAXIMA100	1033	824	1096	249	603
EQUALMAXIMA150	1090	961	1215	279	714
EQUALMAXIMA200	1184	988	1356	279	741
EQUALMAXIMA250	1235	1054	1371	294	794
EXPONENTIAL50	727	481	2402	254	59
EXPONENTIAL100	590	285	883	285	64
EXPONENTIAL150	335	317	317	317	70
EXPONENTIAL200	336	325	323	325	78
EXPONENTIAL250	355	340	339	339	80
GRIEWANKROSENBROCK50	1119	813	1598	711	370
GRIEWANKROSENBROCK100	1084	886	990	870	503
GRIEWANKROSENBROCK150	1237	1054	1088	1030	513
GRIEWANKROSENBROCK200	1265	1208	1234	1187	704
GRIEWANKROSENBROCK250	1359	1264	1267	1239	856
ROSENBROCK50	922	632	1546	592	1896
ROSENBROCK100	1102	855	971	841	2510
ROSENBROCK150	1313	1123	1136	1104	3116
ROSENBROCK200	1493	1339	1362	1314	3430
ROSENBROCK250	1731	1594	1601	1568	3628
SHARPRIDGE50	830	503	2443(0.96)	379	275(0.96)
SHARPRIDGE100	1741	424(0.96)	1789	419(0.96)	267
SHARPRIDGE150	630	434(0.96)	1128(0.96)	427(0.96)	259
SHARPRIDGE200	697	438	867	437	291
SHARPRIDGE250	569(0.93)	422(0.96)	841	426(0.96)	265(0.96)
SINUSOIDAL50	529(0.1)	441(0)	490(0)	436(0)	224(0)
SINUSOIDAL100	747(0)	370(0)	801(0)	370(0)	149(0)
SINUSOIDAL150	288(0)	269(0)	269(0)	269(0)	22(0)
SINUSOIDAL200	288(0)	269(0)	269(0)	269(0)	22(0)
SINUSOIDAL250	303(0)	283(0)	283(0)	283(0)	23(0)
ZAKHAROV50	483	323	475	352	325
ZAKHAROV100	604	416	693	413	487
ZAKHAROV150	713	487	817	504	486
ZAKHAROV200	750	528	821	579	663
ZAKHAROV250	885	538	893	693	501
TOTAL	46,127(0.9)	33,013(0.89)	59,988(0.89)	29,362(0.89)	41,759(0.89)

Table 7 presents the comparative performance of the ECHO+APP+MEM method against other modern optimization techniques on high-dimensional problems (ranging from 50 to 250 dimensions), where the population size is determined dynamically according to the formula: $Np = 4 + \lfloor 3 \cdot \log(\text{dimension}) \rfloor$. The first row lists the methods under comparison: CLPSO, jDE, SaDE, CMA-ES, and the ECHO+APP+MEM method. The first column contains the test functions used in the evaluation, while the table entries report the number of objective function calls required to reach an acceptable solution. In some cases, the success rate is also indicated in parentheses, showing the percentage of runs in which the global minimum was successfully found.

The overall performance of the ECHO+APP+MEM method is highly satisfactory, with a total of 41,759 calls and an average success rate of 89%, comparable to the other methods. jDE and CMA-ES achieve slightly fewer total calls (33,013 and 29,362 respectively), but in many individual cases the ECHO+APP+MEM method converges more quickly. For example, in the EXPONENTIAL150 to EXPONENTIAL250 functions, the ECHO+APP+MEM method requires only 70 to 80 calls, outperforming or matching the most efficient competitors. A similar trend is seen in the SINUSOIDAL functions, where all methods exhibit low success rates, yet the ECHO+APP+MEM method consistently requires far fewer calls only 22 or 23, compared to 269 and above for the others.

In the SHARPRIDGE functions, the ECHO+APP+MEM method shows very strong performance with low function evaluations and high success rates. For instance, in SHARPRIDGE250, it needs only 265 calls with a 96% success rate, while jDE and CMA-ES require more than 420 calls. This suggests an effective handling of rugged landscapes with many local minima and abrupt variations.

However, in more demanding functions such as ROSENBROCK and ELLIPSOIDAL, the ECHO+APP+MEM method shows higher computational cost compared to its competitors. In ROSENBROCK250, for example, it requires 3628 calls, while jDE and CMA-ES achieve similar results in approximately 2000 calls. A similar pattern appears in ELLIPSOIDAL250, where the ECHO+APP+MEM method requires 2790 calls compared to roughly 1340 for CMA-ES. This indicates that for problems characterized by high curvature or strong inter-variable dependencies, the ECHO+APP+MEM method may require more iterations to converge.

Overall, the results in this table suggest that the ECHO+APP+MEM method is particularly effective in problems with broad attraction basins, many local optima, or non-convex structures, as it manages to reach good solutions with relatively few function evaluations. In contrast, it may need more effort for convergence in problems with pronounced curvature. Nevertheless, its high success rate and the consistent stability across various problem types make it a competitive and robust choice for high-dimensional optimization tasks.

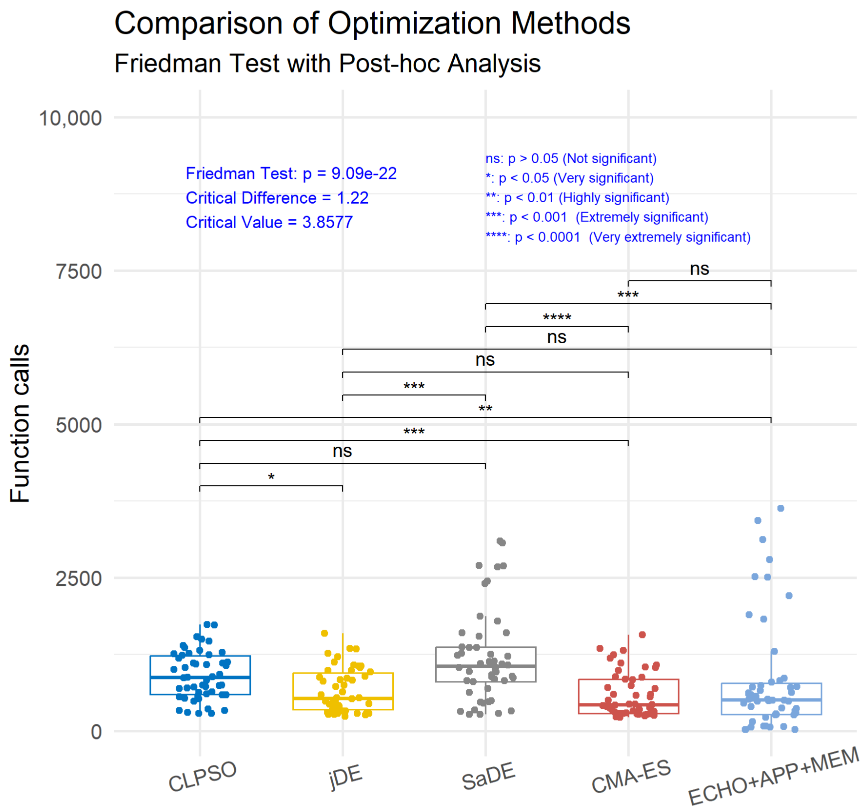
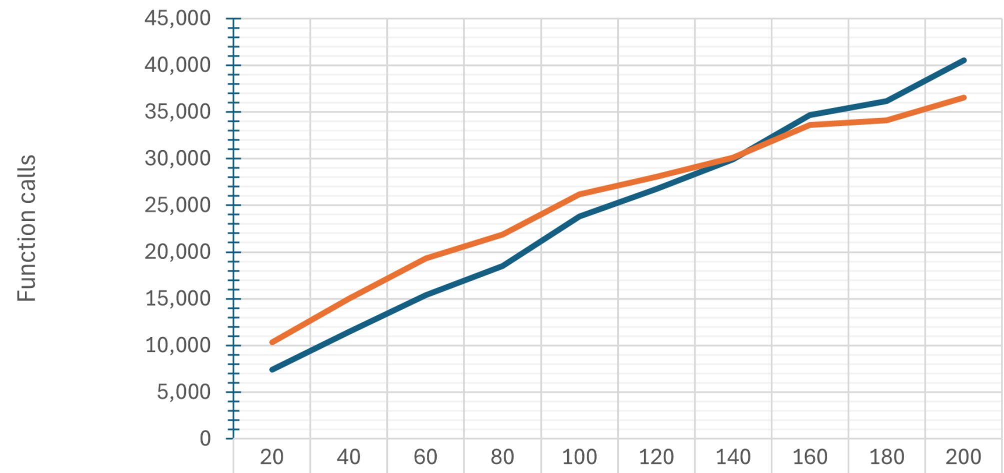


Figure 3. Statistical comparison of function calls of the different optimization methods for high-dimensional benchmark functions (population: formula)

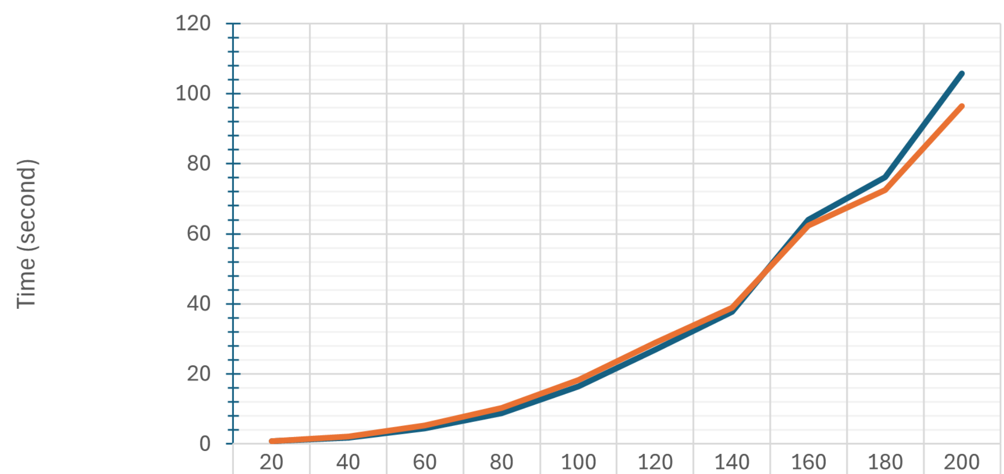
In Figure 3 the comparative analysis between the ECHO+APP+MEM method and other well-known optimization methods showed significant results based on the Friedman test, where the p-value is extremely small (9.09e-22), indicating statistically significant differences among the methods. The critical difference is 1.22 and the critical value is 3.85. The ECHO+APP+MEM method demonstrated statistically significant superiority compared to CLPSO ($p<0.01$) and SaDE ($p<0.001$), while there were no statistically significant differences compared to jDE and CMA-ES. Overall, the ECHO+APP+MEM method appears to perform better or at least equivalently compared to most of the methods tested, which reinforces its reliability and effectiveness.



	20	40	60	80	100	120	140	160	180	200
ELP	7385	11,434	15,402	18,502	23,806	26,760	29,915	34,623	36,170	40,501
ROSENBROCK	10,363	15,001	19,319	21,883	26,189	28,067	30,085	33,617	34,075	36,530

Figure 4. Computational performance (Function calls) of the ECHO+APP+MEM method on ELP and ROSENBROCK across dimensions 20-200

Figure 4 illustrates the complexity of the ECHO+APP+MEM method in relation to the problem dimensionality, which ranges from 20 to 200. The values represent the number of objective function calls for the ELP and ROSENBROCK functions. It is observed that as dimensionality increases, the number of function calls rises significantly for both functions, indicating that the complexity of the method grows with the problem size. For ELP, the calls increase from 7385 at dimension 20 to 40,501 at dimension 200, while for ROSENBROCK, the calls increase from 10,363 to 36,530 respectively. This shows that the method requires more computations for higher dimensions, which is expected for complex optimization problems.



	20	40	60	80	100	120	140	160	180	200
ELP	0.75	1.82	4.45	8.73	16.48	27.01	37.80	64.08	76.11	105.76
ROSENBROCK	0.78	2.20	5.33	10.31	18.26	28.95	38.95	62.31	72.56	96.50

Figure 5. Computational performance (Time) of the ECHO+APP+MEM method on ELP and ROSENBROCK across dimensions 20-200

Figure 5 shows the complexity of the ECHO+APP+MEM method based on the solving time in seconds for the same functions and dimensionalities. The solving time also increases with dimensionality, starting from less than one second at dimension 20 and reaching approximately 106 seconds for ELP and 96.5 seconds for ROSENBROCK at dimension 200.

The increase in time is steady and corresponds to the increase in the number of objective function calls, confirming that the method's complexity is linked to the dimensionality and the computational effort required to solve the problems. Overall, both figures clearly show that the ECHO+APP+MEM method remains functional and scalable but with increasing computational cost as the problem dimension grows.

3.3. Experiments on Real-World Problems

In this section, experiments were conducted with a focus on real-world optimization problems, selected from the set of functions listed in Table 8, which represent approximately 85% of the entire CEC2011 benchmark set and encompass both classical and contemporary challenges in global optimization. The experimental procedure was carefully designed to ensure strict comparability among the different algorithms under study. Specifically, a fixed population size of 100 samples, generated with a uniform distribution within the feasible domain of each problem, was used for all methods except CMA-ES. For CMA-ES, standard practice from the relevant literature was followed, setting the population size according to the formula $Np = 4 + \lfloor 3 \cdot \log(\text{dimension}) \rfloor$ to optimally leverage the algorithm's dynamics relative to problem dimensionality. It is noteworthy that no local optimization phase was applied in any experiment, thus allowing the pure global search capability of each algorithm to be evaluated without the assistance of local refinement procedures. All other parameters governing the operation of the algorithms including evolution coefficients, stopping rules, and acceptance criteriaremained unchanged and were set according to the detailed configurations presented in Table 1, thereby ensuring maximum objectivity in the comparative assessment of results. Through this systematic approach, the experimental outcomes faithfully reflect the behavior of the methods on demanding real-world problems, enabling reliable conclusions to be drawn regarding their practical effectiveness.

The analysis of the results presented in Tables 9, 10, and 11 was based on a rigorously standardized experimental protocol to ensure the reliability and comparability of the methods. Specifically, for Table 9, the termination criterion was strictly set to a fixed number of function evaluations (1,500,000), providing all methods with equal opportunities for exploration in the solution space. For each method, the best value (i.e., the lowest objective function value found) and the mean value over 30 independent runs were recorded, thus capturing both the maximum and average performance in a statistically substantiated manner. Subsequently, in Table 10, the methods were ranked for each problem according to their performance, with 1 corresponding to the best and 5 to the least effective performance among the five methods. For drawing overall conclusions, Table 11 presents the total sum of the ranks obtained by each method for both the best and mean values, and this aggregate score was divided by the total number of positions (i.e., the number of methods multiplied by two, to account for both criteria), providing a composite measure for comparing overall performance. Thus, the smallest average total score indicates the most effective method, enabling an objective evaluation of the comparative superiority of each approach across the full set of problems.

Table 8. Real world problems CEC2011.

PROBLEM	FORMULA	Dim	BOUNDS
Parameter Estimation for Frequency-Modulated Sound Waves	$\min_{x\in[-6.4,6.35]^6}\ f(x) = \frac{1}{N}\sum_{n=1}^{N}\lvert y(n;x) - y_{\text{target}}(n)\rvert^2$ $y(n;x) = x_0\sin(x_1 n + x_2\sin(x_3 n + x_4\sin(x_5 n)))$	6	$x_i \in [-6.4, 6.35]$
Lennard-Jones Potential	$\min_{x\in\mathbb{R}^{3N-6}}\ f(x) = 4\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\left[\left(\frac{1}{r_{ij}}\right)^{12} - \left(\frac{1}{r_{ij}}\right)^{6}\right]$	30	$x_0 \in (0,0,0)$ $x_1, x_2 \in [0,4]$ $x_3 \in [0,\pi]$ x_{3k-3} x_{3k-2} $x_i \in [-b_k, +b_k]$
Bifunctional Catalyst Blend Optimal Control	$\frac{dx_1}{dt} = -k_1x_1,\ \frac{dx_2}{dt} = k_1x_1 - k_2x_2 + k_3x_2 + k_4x_3,$ $\frac{dx_3}{dt} = k_2x_2,\ \frac{dx_4}{dt} = -k_4x_4 + k_5x_5,$ $\frac{dx_5}{dt} = -k_3x_2 + k_6x_4 - k_5x_5 + k_7x_6 + k_8x_7 + k_9x_5 + k_{10}x_7$ $\frac{dx_6}{dt} = k_8x_5 - k_7x_6,\ \frac{dx_7}{dt} = k_9x_5 - k_{10}x_7$ $k_i(u) = c_{i1} + c_{i2}u + c_{i3}u^2 + c_{i4}u^3$	1	$u \in [0.6, 0.9]$
Optimal Control of a Non-Linear Stirred Tank Reactor	$J(u) = \int_0^{0.72}\left[x_1(t)^2 + x_2(t)^2 + 0.1u^2\right]dt$ $\frac{dx_1}{dt} = -2x_1 + x_2 + 1.25u + 0.5\exp\left(\frac{x_1}{x_1+2}\right)$ $\frac{dx_2}{dt} = -x_2 + 0.5\exp\left(\frac{x_1}{x_1+2}\right)$ $x_1(0) = 0.9,\quad x_2(0) = 0.09,\ t\in[0,0.72]$	1	$u \in [0,5]$
Tersoff Potential for model Si (B)	$\min_{\mathbf{x}\in\Omega} f(\mathbf{x}) = \sum_{i=1}^{N} E(\mathbf{x}_i)$ $E(\mathbf{x}_i) = \frac{1}{2}\sum_{j\neq i} fc(r_{ij})\left[V_R(r_{ij}) - B_{ij}V_A(r_{ij})\right]$ where $r_{ij} = \lVert\mathbf{x}_i - \mathbf{x}_j\rVert$, $V_R(r) = A\exp(-\lambda_1 r)$ $V_A(r) = B\exp(-\lambda_2 r)$ $f_C(r)$: cutoff function with $f_C(r)$: angle parameter	30	$x_1 \in [0,4]$ $x_2 \in [0,4]$ $x_3 \in [0,\pi]$ $x_i \in \left[\frac{4(i-3)}{4}, 4\right]$
Tersoff Potential for model Si (C)	$\min_{\mathbf{x}}\ V(\mathbf{x}) = \sum_{i=1}^{N}\sum_{j>i}^{N} f_C(r_{ij})\left[a_{ij}f_R(r_{ij}) + b_{ij}f_A(r_{ij})\right]$ $f_C(r) = \begin{cases} 1, & r < R - D \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{\pi(r-R+D)}{2D}\right), & \lvert r - R\rvert \leq D \\ 0, & r > R + D \end{cases}$ $f_R(r) = A\exp(-\lambda_1 r)$ $f_A(r) = -B\exp(-\lambda_2 r)$ $b_{ij} = \left[1 + (\beta^n)\zeta_{ij}^n\right]^{-1/(2n)}$ $\sum_{k\neq i,j} f_C(r_{ik})g(\theta_{ijk})\exp\left[\lambda_3^3(r_{ij} - r_{ik})^3\right]$	30	$x_1 \in [0,4]$ $x_2 \in [0,4]$ $x_3 \in [0,\pi]$ $x_i \in \left[\frac{4(i-3)}{4}, 4\right]$
Spread Spectrum Radar Polly phase Code Design	$\min_{x\in X}\ f(x) = \max\{\lvert\varphi_1(x)\rvert, \lvert\varphi_2(x)\rvert, \ldots, \lvert\varphi m(x)\rvert\}$ $X = \{x \in \mathbb{R}^n \mid 0 \leq x_j \leq 2\pi,\ j = 1,\ldots,n\} m = 2n - 1$ $\varphi_j(x) = \begin{cases} \sum_{k=1}^{n-j}\cos(x_k - x_{k+j}) & \text{for } j = 1,\ldots,n-1 \\ n & \text{for } j = n \\ \varphi_{2n-j}(x) & \text{for } j = n+1,\ldots,2n-1 \end{cases}$ $\varphi_j(x) = \sum_{k=1}^{n-j}\cos(x_k - x_{k+j}),\quad j = 1,\ldots,n-1$ $\varphi n(x) = n,\ \varphi_{n+\ell}(x) = \varphi_{n-\ell}(x),\quad \ell = 1,\ldots,n-1$	20	$x_j \in [0, 2\pi]$
Transmission Network Expansion Planning	$\min\sum_{l\in\Omega} c_l n_l + W_1\sum_{l\in OL}\lvert f_l - \bar{f}_l\rvert + W_2\sum_{l\in\Omega}\max(0, n_l - \bar{n}_l)$ $Sf = g - d$ $f_l = \gamma_l n_l \Delta\theta_l,\quad \forall l \in \Omega$ $\lvert f_l\rvert \leq \bar{f}_l n_l,\quad \forall l \in \Omega$ $0 \leq n_l \leq \bar{n}_l,\quad n_l \in \mathbb{Z},\quad \forall l \in \Omega$	7	$0 \leq n_i \leq \bar{n}_l$ $n_i \in \mathbb{Z}$
Electricity Transmission Pricing	$\min_x\ f(x) = \sum_{i=1}^{N_g}\left(\frac{C_i^{gen}}{P_i^{gen}} - R_i^{gen}\right)^2 + \sum_{j=1}^{N_d}\left(\frac{C_j^{load}}{P_j^{load}} - R_j^{load}\right)^2$ $\sum_j GD_{i,j} + \sum_j BT_{i,j} = P_i^{gen},\quad \forall i$ $\sum_i GD_{i,j} + \sum_i BT_{i,j} = P_j^{load},\quad \forall j$ $GD_{i,j}^{max} = \min(P_i^{gen} - BT_{i,j},\ P_j^{load} - BT_{i,j})$	126	$GD_{i,j} \in [0, GD_{i,j}^{max}]$
Circular Antenna Array Design	$\min_{r_1,\ldots,r_6,\varphi_1,\ldots,\varphi_6}\quad f(\mathbf{x}) = \max_{\theta\in\Omega} AF(\mathbf{x},\theta)$ $AF(\mathbf{x},\theta) = \left\lvert\sum_{k=1}^{6}\exp\left(j\left[2\pi r_k\cos(\theta - \theta_k) + \varphi_k\frac{\pi}{180}\right]\right)\right\rvert$	12	$r_k \in [0.2, 1]$ $\varphi_k \in [-180, 180]$
Dynamic Economic Dispatch 1	$\min_{\mathbf{P}}\quad f(\mathbf{P}) = \sum_{t=1}^{24}\sum_{i=1}^{5}\left(a_iP_{i,t}^2 + b_iP_{i,t} + c_i\right)$ $P_i^{\min} \leq P_{i,t} \leq P_i^{\max},\quad \forall i = 1,\ldots,5,\ t = 1,\ldots,24$ $\sum_{i=1}^{5} P_{i,t} = D_t,\quad \forall t = 1,\ldots,24$ $P_{\min} = [10, 20, 30, 40, 50]$ $P_{\max} = [75, 125, 175, 250, 300]$	120	$P_i^{\min} \leq P_{i,t} \leq P_i^{\max}$
Dynamic Economic Dispatch 2	$\min_{\mathbf{P}}\quad f(\mathbf{P}) = \sum_{t=1}^{24}\sum_{i=1}^{9}\left(a_iP_{i,t}^2 + b_iP_{i,t} + c_i\right)$ $P_i^{\min} \leq P_{i,t} \leq P_i^{\max},\quad \forall i = 1,\ldots,5,\ t = 1,\ldots,24$ $\sum_{i=1}^{5} P_{i,t} = D_t,\quad \forall t = 1,\ldots,24$ $P_{\min} = [150, 135, 73, 60, 73, 57, 20, 47, 20]$ $P_{\max} = [470, 460, 340, 300, 243, 160, 130, 120, 80]$	216	$P_i^{\min} \leq P_{i,t} \leq P_i^{\max}$
Static Economic Load Dispatch (1,2,3,4,5)	$\min_{P_1,\ldots,P_{N_G}}\ F = \sum_{i=1}^{N_G} f_i(P_i)$ $f_i(P_i) = a_iP_i^2 + b_iP_i + c_i,\quad i = 1,2,\ldots,N_G$ $f_i(P_i) = a_iP_i^2 + b_iP_i + c_i + \lvert e_i\sin(f_i(P_i^{\min} - P_i))\rvert$ $P_i^{\min} \leq P_i \leq P_i^{\max},\quad i = 1,2,\ldots,N_G$ $\sum_{i=1}^{N_G} P_i = P_D + P_L$ $P_L = \sum_{i=1}^{N_G}\sum_{j=1}^{N_G} P_iB_{ij}P_j + \sum_{i=1}^{N_G} B_{0i}P_i + B_{00}$ $P_i - P_i^0 \leq UR_i\quad P_i^0 - P_i \leq DR_i$	6 13 15 40 140	See Technical Report of CEC2011

Table 9. Algorithms' Comparison Based on Best and Mean after 1.5e+5 FEs.+

PROBLEM	CLPSO best	CLPSO Mean	SaDE best	SaDE Mean	jDE Best	jDE Mean	CMA-ES best	CMA-ES Mean	EO Best	EO Mean
Parameter Estimation for Frequency-Modulated Sound Waves	0.1314837477	0.2124981688	0.1899428536	0.2025566839	0.2915779318	0.372177177	0.18160916	0.256863966	0.2521750203	0.2954215036
Lennard-Jones Potential	-13.43649135	-10.25073403	-24.86870825	-22.6693403	-26.96868295	-23.05977012	-28.42253189	-25.78783328	-15.93299933	-12.14897801
BifunctionalCatalyst Blend Optimal Control	0.2374871493	0.4314871688	0.545953325	0.624508763	0.545624975	0.6003624983	0.1314837477	0.2445629767	0.245874592	0.3935178477
Optimal Control of a Non-Linear Stirred Tank Reactor	0.3903767228	0.3903767228	0.3903767228	0.3903767228	0.4913066735	7.309621249e+10	0.3903767228	0.390376723	38.94304848	26682.93618
Tersoff Potential for model Si (B)	-28.23544117	-26.18834522	-3.107773136	25.4711091	-22.10581388	-2.387797305	-29.26244222	-27.5889735	-26.7539495	-24.83294286
Tersoff Potential for model Si (C)	-30.85200257	-28.87349048	-11.60719468	22.08963599	-23.1614585	-7.886665429	-33.19699356	-31.79270914	-31.49334757	-28.7003262
Spread Spectrum Radar Polly phaseCode Design	1.085334991	1.343956153	1.536501579	2.150881715	0.9189283507	1.616525419	0.014848227	0.171988666	0.5931349058	0.8611681213
Transmission Network Expansion Planning	250	250	250	250	250	250	250	250	250	250
Electricity Transmission Pricing	13775010.1	13775395.07	23481009.86	30034934.81	13776257.77	14484728.07	13775841.77	13787550.18	13776506.06	13777507.07
Circular Antenna Array Design	0.006933401045	0.05181551798	0.02142329927	0.03892428051	0.04238306654	0.1610286721	0.007204797576	0.008635655364	0.03477651401	0.1675322066
Dynamic Economic Dispatch 1	428607927.6	435250914.5	968042312.1	1034679775	968042312.1	1071143842	88285.6024	102776.7103	473972051.7	496720571.7
Dynamic Economic Dispatch 2	33031590.31	53906147.38	845287898.3	913715793.2	350885523	456293948.4	502699.4187	477720.1511	77187145.39	92911983.83
Static Economic Load Dispatch 1	6554.672173	7668.333603	16877.92071	101588.3887	6473.96039	939805.3726	6657.613028	415917.4625	6878.581584	12522.32104
Static Economic Load Dispatch 2	19030.36081	20699.00219	2600565.214	9329466.813	41350.66254	8649652.933	763001.2185	1425815.44	19406.19272	44368.22458
Static Economic Load Dispatch 3	470192288.3	470294703.2	478069615.3	541898763	474743051.8	514454706.1	470023232.3	470023232.3	470437730.6	470713100.1
Static Economic Load Dispatch 4	884980.5569	1423887.358	14170362.58	106749078.5	14657102.04	32906828.34	476053.5197	2925852.935	149544.9615	417426.7844
Static Economic Load Dispatch 5	8105947615	8110924071	8901002004	9943082701	8446155955	463055675	8072077963	8084017791	8119017656	8263919973

Table 10. Algorithms' Comparison Based on Best and Mean after 1.5e+5 FEs.

PROBLEM	CLPSO best	CLPSO Mean	SaDE best	SaDE Mean	jDE Best	jDE Mean	CMA-ES best	CMA-ES Mean	EO Best	EO Mean
Parameter Estimation for Frequency-Modulated Sound Waves	1	2	3	1	5	5	2	3	4	4
Lennard-Jones Potential	5	5	3	3	2	2	1	1	4	4
BifunctionalCatalyst Blend Optimal Control	2	2	5	5	4	4	1	1	3	3
Optimal Control of a Non-Linear Stirred Tank Reactor	1	1	1	1	3	3	1	1	4	4
Tersoff Potential for model Si (B)	2	2	5	5	4	4	1	1	3	3
Tersoff Potential for model Si (C)	2	2	5	5	4	4	1	1	3	3
Spread Spectrum Radar Polly phaseCode Design	4	3	5	5	3	4	1	1	2	2
Transmission Network Expansion Planning	1	1	1	1	1	1	1	1	1	1
Electricity Transmission Pricing	2	1	5	5	1	4	3	3	4	2
Circular Antenna Array Design	1	3	3	2	5	4	2	1	4	5
Dynamic Economic Dispatch 1	2	2	4	4	4	5	1	1	3	3
Dynamic Economic Dispatch 2	2	2	5	5	4	4	1	1	3	3
Static Economic Load Dispatch 1	2	1	5	3	1	5	3	4	4	2
Static Economic Load Dispatch 2	1	1	3	5	4	4	5	3	2	2
Static Economic Load Dispatch 3	2	2	5	5	4	4	1	1	3	3
Static Economic Load Dispatch 4	3	2	4	5	5	4	2	3	1	1
Static Economic Load Dispatch 5	2	2	5	5	4	4	1	1	3	3
TOTAL	**35**	**34**	**68**	**66**	60	68	**28**	**28**	**54**	51

Table 11. Comparison of algorithms and final ranking

Problem	Best	Mean	Overall	Average	Rang
CMA-ES	28	28	56	1.647	**1**
CLPSO	35	34	69	2.029	**2**
EO	51	48	99	2.9116	**3**
jDE	58	65	123	3.617	**4**
SaDE	67	65	132	3.882	**5**

In the comparative assessment conducted in this subsection, the basic form of the EO was benchmarked against other modern and internationally recognized evolutionary algorithms such as SaDE, jDE, CLPSO, and CMA-ES, which serve as the state-of-the-art reference for real-world global optimization problems. The selection of these algorithms ensures a fair and realistic comparison, as all are noted for their stability and proven effectiveness in a wide range of practical applications and complex problems, as also reflected in the CEC2011 benchmark. The EO was implemented in its pure form, without the use of additional memory or surrogate evaluation mechanisms, thus highlighting the intrinsic ability of the method to address high-dimensional and complex problems transparently.

Analysis of the results demonstrates that EO remains highly competitive, and in many cases superior, to the other examined methods across a substantial subset of the tested problems. EO often achieves lower or comparable best and mean values, as well as reduced dispersion between independent runs a feature indicative of stable performance and reliability in real-world scenarios. Although in certain instances minor deviations are observed in favor of competing algorithms, especially in highly specialized problem landscapes or regions favoring intensive local exploitation, the overall picture confirms the effectiveness of EO. The uniform use of a fixed population size (100 samples, except for CMA-ES), identical stopping criteria and parameters (as specified in Table 1), and the absence of local optimization ensure absolute objectivity and fairness in the evaluation framework.

Consequently, EO proves to be a highly promising alternative among contemporary global optimization algorithms, highlighting its potential for future development and adoption in real-world applications, where the balance between speed, reliability, and implementation simplicity remains a core requirement.

3.4. The Echo method and neural networks

An additional experiment was conducted, where the ECHO+APP+MEM method was used to train artificial neural networks [53,54], by minimizing the so - called training error defined as:

$$E\left(N\left(\overrightarrow{x},\overrightarrow{w}\right)\right)=\sum_{i=1}^{M}\left(N\left(\overrightarrow{x}_i,\overrightarrow{w}\right)-y_i\right)^2 \tag{9}$$

In this equation the function $N\left(\overrightarrow{x},\overrightarrow{w}\right)$ represents the artificial neural network which is applied on a vector $\overrightarrow{x}$ and the vector $\overrightarrow{w}$ denotes the parameter vector of the neural network. The set $\left(\overrightarrow{x_i},y_i\right),\ i=1,...,M$ represents the training set of the objective problem and the values y_i are the expected outputs for each pattern $\overrightarrow{x_i}$.

To validate the ECHO+APP+MEM method, an extensive collection of classification datasets was employed, sourced from various publicly available online repositories. These datasets were obtained from:

1. The UCI database, https://archive.ics.uci.edu/(accessed on 5 July 2025)[55]
2. The Keel website, https://sci2s.ugr.es/keel/datasets.php(accessed on 5 July2025)[56].
3. The Statlib URL https://lib.stat.cmu.edu/datasets/index(accessed on 5 July 2025).

The experiments were conducted using the following datasets:

1. **Appendictis** which is a medical dataset [57].

2. **Alcohol**, which is dataset regarding alcohol consumption [58].
3. **Australian**, which is a dataset produced from various bank transactions [59].
4. **Balance** dataset [60], produced from various psychological experiments.
5. **Circular** dataset, which is an artificial dataset.
6. **Cleveland**, a medical dataset which was discussed in a series of papers [61,62].
7. **Dermatology**, a medical dataset for dermatology problems [63].
8. **Ecoli**, which is related to protein problems [64].
9. **Glass** dataset, that contains measurements from glass component analysis.
10. **Haberman**, a medical dataset related to breast cancer.
11. **Hayes-roth** dataset [65].
12. **Heart**, which is a dataset related to heart diseases [66].
13. **HeartAttack**, which is a medical dataset for the detection of heart diseases
14. **Housevotes**, a dataset which is related to the Congressional voting in USA [67].
15. **Ionosphere**, a dataset that contains measurements from the ionosphere [68,69].
16. **Liverdisorder**, a medical dataset that was studied thoroughly in a series of papers[70, 71].
17. **Lymography** [72].
18. **Mammographic**, which is a medical dataset used for the prediction of breast cancer [73].
19. **Parkinsons**, which is a medical dataset used for the detection of Parkinson's disease [74,75].
20. **Pima**, which is a medical dataset for the detection of diabetes[76].
21. **Popfailures**, a dataset related to experiments regarding climate [77].
22. **Regions2**, a medical dataset applied to liver problems [78].
23. **Saheart**, which is a medical dataset concerning heart diseases[79].
24. **Segment** dataset [80].
25. The **Sonar** dataset, related to sonar signals [81].
26. **Statheart**, a medical dataset related to heart diseases.
27. **Student**, which is a dataset regarding experiments in schools [82].
28. **Transfusion**, which is a medical dataset [83].
29. **Wdbc**, which is a medical dataset regarding breast cancer [84,85].
30. **Wine**, a dataset regarding measurements about the quality of wines [86,87].
31. **EEG**, which is dataset regardingEEG recordings [88,89]. From this dataset the following cases were used: Z_F_S, ZO_NF_S, and ZONF_S.
32. **Zoo**, which is a dataset regarding animal classification [90] .

DATASET	GENETIC	ADAM	NEAT	PRUNE	ECHO	ECHO+APP	ECHO +MEM	ECHO +APP+MEM
Appendicitis	18.10%	16.50%	17.20%	15.97%	22.07%	22.47%	22.60%	22.77%
Alcohol	39.57%	57.78%	66.80%	15.75%	18.91%	17.24%	17.77%	16.96%
Australian	32.10%	35.65%	31.98%	43.66%	22.34%	22.34%	22.91%	23.16%
Balance	8.97%	12.27%	23.14%	9.00%	7.10%	7.10%	7.22%	7.23%
Circular	5.99%	19.95%	35.18%	12.76%	4.48%	4.48%	4.37%	4.37%
Cleveland	51.60%	67.55%	53.44%	51.48%	45.74%	45.74%	46.60%	46.60%
Dermatology	30.58%	26.14%	35.18%	12.76%	9.34%	9.34%	11.51%	11.51%
Ecoli	54.67%	64.43%	43.44%	60.32%	45.89%	45.89%	48.94%	48.94%
Fert	28.50%	23.98%	15.37%	25.20%	24.40%	24.13%	26.50%	24.57%
Glass	58.30%	61.38%	55.71%	66.19%	50.24%	50.24%	49.43%	49.22%
Haberman	28.66%	29.00%	24.04%	29.38%	28.71%	28.71%	28.80%	28.80%
Hayes-roth	56.18%	59.70%	50.15%	45.44%	35.21%	35.21%	37.46%	37.46%
Heart	28.34%	38.53%	39.27%	27.21%	18.10%	18.10%	19.00%	19.02%
HeartAttack	29.03%	45.55%	32.34%	29.26%	20.06%	20.06%	20.15%	20.15%
Housevotes	6.62%	7.48%	10.89%	5.81%	7.58%	7.68%	7.20%	7.10%
Ionosphere	15.14%	16.64%	19.67%	11.32%	14.53%	14.53%	15.03%	15.08%
Liverdisorder	31.11%	41.53%	30.67%	49.72%	31.82%	31.82%	32.48%	32.48%
LYMOGRAPHY	28.42%	39.79%	33.70%	22.02%	24.60%	24.60%	26.02%	25.31%
Mammographic	19.88%	46.25%	22.85%	38.10%	17.44%	17.45%	17.36%	17.36%
Parkinsons	18.05%	24.06%	18.56%	22.12%	15.04%	14.95%	15.74%	15.21%
Pima	32.19%	34.85%	34.51%	35.08%	26.34%	26.34%	26.78%	26.78%
Popfailures	5.94%	5.18%	7.05%	4.79%	6.88%	6.88%	6.84%	6.84%
Regions2	29.39%	29.85%	33.23%	34.26%	28.21%	28.21%	30.07%	30.12%
Saheart	34.86%	34.04%	34.51%	37.70%	32.71%	32.91%	32.46%	32.46%
Segment	57.72%	49.75%	66.72%	60.40%	14.82%	14.82%	16.91%	16.87%
Sonar	22.40%	30.33%	34.10%	23.80%	19.98%	20.52%	20.37%	20.07%
Spiral	48.66%	47.67%	48.66%	50.38%	41.91%	41.91%	42.24%	42.24%
STATHEART	27.25%	44.04%	44.36%	28.37%	19.45%	19.45%	20.24%	20.24%
Student	5.61%	5.13%	10.20%	10.84%	5.84%	5.84%	5.98%	5.98%
Transfusion	24.87%	25.68%	24.87%	29.35%	23.70%	23.70%	24.09%	24.09%
Wdbc	8.56%	35.35%	12.88%	15.48%	3.92%	3.92%	4.72%	4.60%
Wine	19.20%	29.40%	25.43%	16.62%	10.45%	10.45%	12.84%	13.88%
Z_F_S	10.73%	47.81%	38.41%	17.91%	8.48%	8.48%	8.71%	7.89%
ZO_NF_S	21.54%	47.43%	43.75%	15.57%	5.15%	5.15%	5.61%	5.99%
ZONF_S	4.36%	11.99%	5.44%	3.27%	2.96%	2.96%	2.91%	2.83%
ZOO	9.50%	14.13%	20.27%	8.53%	3.60%	3.60%	3.60%	3.70%
AVERAGE	**26.46%**	**34.08%**	**31.78%**	**27.38%**	**19.94%**	**19.92%**	**20.60%**	**20.50%**

Table 12. Comparative Classification Error Rates Across Datasets Using ECHO-Based and a series of Optimization Methods

Table 12 presents the classification error rates for a wide range of datasets, evaluated using a series of techniques to train an artificial neural network with 10 processing nodes. These methods include

1. A standard Genetic Algorithm (GENETIC)
2. The column ADAM denotes the usage of the ADAM optimization method [91] in order to train a neural network with $H = 10$ processing nodes.
3. The column NEAT represents the usage of the NEAT method (NeuroEvolution of Augmenting Topologies) [92].
4. The column PRUNE stands for the the usage of OBS pruning method [93], provided by Fast Compressed Neural Networks library [94].
5. The basic version of the ECHO+APP+MEM method
6. Three enhanced variants ECHO combined with approximate evaluations (APP), memory (MEM), and both (APP+MEM). Each value in the table corresponds to the percentage error for the respective dataset and method. The final row provides the average error rate across all datasets for each method, serving as an overall performance indicator. Lower values indicate better classification accuracy.

The comparative analysis of the error rates presented in this table clearly highlights the overall superiority of the ECHO and ECHO+APP methods compared to both classic evolutionary algorithms and the other variants of the same family. The average error rates for these two methods are particularly low (19.94% for ECHO and 19.92% for ECHO+APP), outperforming not only the Genetic Algorithm (26.46%) but also the ECHO variants that include memory mechanisms (20.60% and 20.50%).

The consistently improved performance of ECHO and ECHO+APP demonstrates the strength of the core algorithm in maintaining a balance between exploration and exploitation, while simultaneously offering adaptability to a wide range of datasets and problem types. The integration of the approximate evaluations technique in ECHO+APP plays a significant role in reducing computational cost without a substantial loss in accuracy, as evidenced by the minimal difference in average error between ECHO and ECHO+APP. This result shows that the practical advantage of faster fitness estimation does not come at the expense of classification quality.

In contrast, the use of memory mechanisms (MEM), whether combined with or without approximate evaluations, appears to introduce additional complexity that limits the algorithm's flexibility and adaptability to heterogeneous datasets. While in certain specific problems, such as Z_F_S, memory-based variants achieve marginally better results, their overall performance remains slightly inferior.

In major benchmark datasets such as Alcohol, Dermatology, Segment, and Wine, ECHO+APP achieves the lowest error rates, demonstrating its stability and reliability even in challenging classification scenarios. The fact that the simplest variants consistently remain at the top, often outperforming the more complex ones, confirms the effectiveness and practical utility of ECHO and ECHO+APP, especially in machine learning applications of small to moderate complexity.

In conclusion, the results summarized in the table provide strong evidence that ECHO and ECHO+APP combine high accuracy with low computational overhead, offering a robust and efficient solution for a wide range of classification problems, with significant potential for practical deployment in real-world scenarios.

4. Conclusions

The Echo Optimizer, inspired by the natural behavior of sound echoes, which utilizes original mechanisms such as echoic memory and approximate evaluation for the effective resolution of complex problems. The algorithm is designed to balance the exploration of new regions within the search space and the exploitation of the best-known solutions, offering a flexible and adaptive approach to challenges that involve significant computational complexity. Its innovative mechanisms, including echo memory and approximate evaluation, enable efficient management of computational resources without compromising accuracy or convergence speed.

Experimental evaluations highlight the competitive, and even superior, performance of the Echo Optimizer across a wide range of benchmark optimization functions. Multimodal functions, characterized by numerous local minima, are a domain where the algorithm excels. The echo memory mechanism, by storing and reusing previously evaluated solutions, significantly reduces the number of required objective function evaluations. This balance allows the algorithm to effectively explore the search space while avoiding redundant computations in already explored regions.

The approximate evaluation mechanism, on the other hand, accelerates the process by identifying and rejecting unpromising solutions before fully evaluating the objective function. This makes the Echo Optimizer highly efficient for functions with broad basins of attraction, where the goal is rapid convergence to the global minimum. Furthermore, the integration of these two techniques ensures that solutions stored in memory are evaluated accurately, enhancing the reliability of the algorithm. While high-dimensional problems remain challenging, the algorithm maintains robust performance by adjusting its parameters to meet the demands of each specific problem.

The method's effectiveness is further supported by its structural flexibility. Parameters such as the decay factor and reflection coefficient can be fine-tuned to adapt to different problems, while the memory and approximation mechanisms are designed to function either independently or synergistically, optimizing the use of available resources. For instance, in problems with numerous local minima, such as RASTRIGIN and GKLS, the echo memory proves extremely effective. Conversely, in smoother functions, the approximate evaluation accelerates the process by reducing overall execution time.

Moreover, the method adapts to the requirements of complex, high-dimensional problems. While computational demands increase in larger search spaces, the algorithm remains competitive compared to leading state-of-the-art methods such as CMA-ES. Its ability to efficiently manage the complexity of the search space without an exponential increase in computational resources underscores the adaptability of the Echo Optimizer.

In summary, the Echo Optimizer provides a comprehensive optimization framework that combines design simplicity with computational efficiency. Experimental results confirm that the method is ideal for addressing multimodal problems, achieving rapid convergence in smooth functions, and maintaining competitive performance in high-dimensional environments. These characteristics make the algorithm an excellent choice for applications requiring high accuracy, speed, and flexibility, offering significant potential for future research and development in solving complex optimization challenges.

Future Research Directions include several promising avenues:

- Dynamic Parameter Adjustment: Developing mechanisms to dynamically adapt parameters such as reflection and decay coefficients based on the algorithm's behavior during execution.
- Hybrid Approaches: Combining EO with other optimization methods like swarm algorithms or genetic algorithms could further enhance performance, particularly for highly complex problems.
- Real-World Applications: Implementing the algorithm in domains such as energy optimization, supply chain management, and neural network training.
- Theoretical Convergence Analysis: Further investigation of the algorithm's mathematical properties, including convergence rate and probability of escaping local optima.
- Intelligent Adaptation Systems: Incorporating artificial intelligence for automatic adaptation of EO to specific problem characteristics.
- Distributed Computing: Evaluating EO's performance in parallel and distributed systems, especially for large-scale problems.

Continued research on EO promises to further improve its flexibility and effectiveness, expanding its applicability to an even broader range of scientific and industrial problems. These research directions could lead to significant advances in both theoretical understanding and practical applications of this technique.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not applicable.

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

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

1. Tapkin, A. (2023). A Comprehensive Overview of Gradient Descent and its Optimization Algorithms. International Advanced Research Journal in Science, Engineering and Technology, 10 (11), 37-45. DOI: 10.17148/IARJSET.2023.101106
2. Cawade, S., Kudtarkar, A., Sawant, S. & Wadekar, H. (2024). The Newton-Raphson Method: A Detailed Analysis. International Journal for Research in Applied Science & Engineering Technology (IJRASET), 12(11):729-734. DOI: 10.22214/ijraset.2024.65147.
3. Bonate, P.L. (2001). A Brief Introduction to Monte Carlo Simulation. Clinical Pharmacokinetics 40(1):15-22. DOI: 10.2165/00003088-200140010-00002.
4. Eglese, R. W. (1990). Simulated annealing: a tool for operational research. European journal of operational research, 46(3), 271-281.

5. 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
6. Sohail, A. (2023). Genetic algorithms in the fields of artificial intelligence and data sciences. Annals of Data Science, 10(4), 1007-1018.
7. 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.
8. Pant, M., Zaheer, H., Garcia-Hernandez, L., & Abraham, A. (2020). Differential Evolution: A review of more than two decades of research. Engineering Applications of Artificial Intelligence, 90, 103479.
9. Charilogis, V., Tsoulos, I.G.,Tzallas, A., Karvounis, E. (2022). Modifications for the Differential Evolution Algorithm. Symmetry, 2022,14, 447. Doi: https://doi.org/10.3390/sym14030447
10. Charilogis, V.; Tsoulos, I.G. A Parallel Implementation of the Differential Evolution Method. Analytics 2023, 2, 17–30.
11. Vishnoi, N. K. (2021). Algorithms for convex optimization. Cambridge University Press. DOI: https://doi.org/10.1017/9781108699211.016.
12. Vishnoi, N. (2018, May 3). Cutting plane and ellipsoid methods for linear programming (Lecture notes, CS 435, Lecture 9). Yale University. https://nisheethvishnoi.wordpress.com/wp-content/uploads/2018/05/lecture91.pdf
13. Pióro,M. (2004). Routing, Flow, and Capacity Design in Communication and Computer Networks. Chapter 5, 151-210.
14. Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. The Computer Journal, 7(4), 308–313. Doi: https://doi.org/10.1093/comjnl/7.4.308
15. Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. Journal of Global Optimization, 13(4), 455–492. Doi: https://doi.org/10.1023/A:1008306431147
16. Snoek, J., Larochelle, H., & Adams, R. P. (2012).Practical Bayesian optimization of machine learning algorithms.In Advances in Neural Information Processing Systems (NeurIPS 2012), 25, 2951–2959.
17. Mockus, J. (1975). Bayesian approach to global optimization: Theory and applications. Vilnius: Mokslas.
18. Fletcher, R., & Reeves, C. M. (1964).Function minimization by conjugate gradients.The Computer Journal, 7(2), 149–154. Doi: https://doi.org/10.1093/comjnl/7.2.149.
19. Polak, E., & Ribière, G. (1969).Note sur la convergence de méthodes de directions conjuguées.Revue Française d'Informatique et de Recherche Opérationnelle. Série Rouge, 3(16), 35–43.
20. Nocedal, J., & Wright, S. J. (2006). Numerical optimization (2nd ed.). Springer.ISBN: 978-0387303031
21. Benders, J. F. (1962).Partitioning procedures for solving linear programming problems.Numerische Mathematik, 4(1), 238–252.Doi: https://doi.org/10.1007/BF02192511.
22. Geoffrion, A. M., & Dempster, M. A. H. (2021).Benders decomposition in the age of big data and machine learning.Journal of Optimization Theory and Applications, 179(2), 401–425. Doi: https://doi.org/10.1007/s10957-021-01774-5.
23. Dantzig, G. B., & Wolfe, P. (1960).Decomposition principle for linear programs.Operations Research, 8(1), 101–111. Doi: https://doi.org/10.1287/opre.8.1.101.
24. Jones, D. R., & Perttunen, C. D. (1993).Lipschitzian optimization without the Lipschitz constant.Journal of Optimization Theory and Applications, 79(1), 157–181.Doi: https://doi.org/10.1007/BF00941236.
25. Land, A. H., & Doig, A. G. (1960).An automatic method of solving discrete programming problems.Econometrica, 28(3), 497–520. DOI: https://doi.org/10.2307/1907756.
26. Lemarechal, C., & Lasserre, J. (2020).Branch-and-Bound methods in Mixed Integer Nonlinear Programming.Handbook of Discrete Optimization, Elsevier, pp. 297-335. Doi: https://doi.org/10.1016/B978-0-12-801857-3.00009-3
27. 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.
28. Gad, A. G. (2022). Particle swarm optimization algorithm and its applications: a systematic review. Archives of computational methods in engineering, 29(5), 2531-2561
29. Dorigo, M., Maniezzo, V., & Colorni, A. (1996).Ant system: Optimization by a colony of cooperating agents.IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26(1), 29–41. Doi: https://doi.org/10.1109/3477.484436.
30. Rappoport, D., & Schreiber, M. (2021).Recent Advances in Crystal Structure Optimization and Prediction.Crystals, 11(6), 715. Doi: https://doi.org/10.3390/cryst11060715.
31. Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009).GSA: A gravitational search algorithm.Information Sciences, 179(13), 2232–2248. Doi: https://doi.org/10.1016/j.ins.2009.03.004
32. Jang, J. S. R., Sun, C. T., & Mizutani, E. (1997). Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence. Prentice Hall.
33. Zhang, X., & Wu, J. (2010).Photosynthesis algorithm: A new optimization algorithm inspired by natural photosynthesis process.Proceedings of the 2010 International Conference on Artificial Intelligence and Computational Intelligence, 79–83. Doi: https://doi.org/10.1109/AICI.2010.23.
34. Adleman, L. (1994).Towards a mathematical theory of DNA-based computation.DNA Computing, 1, 1–22. Doi: https://doi.org/10.1007/978-1-4615-5890-1_1.
35. Črepinšek, M., Liu, S.-H., Mernik, M., & Ravber, M. (2019). Long Term Memory Assistance for Evolutionary Algorithms. Mathematics, 7(11), 1129. Doi: https://doi.org/10.3390/math7111129

36. Loper, M. (2015). Modeling and Simulation in the Systems Engineering Life Cycle. Surrogate Modeling:Chapter, pp 201–216. SpringerLink, 2015. Doi: https://doi.org/10.1007/978-1-4471-5634-5_1
37. Lam, A. (2020). BFGS in a Nutshell: An Introduction to Quasi-Newton Methods Demystifying the inner workings of BFGS optimization. Towards Data Science.
38. Gianni, A. M., Tsoulos, I. G., Charilogis, V., & Kyrou, G. (2025). Enhancing differential evolution: A dual mutation strategy with majority dimension voting and new stopping criteria. Symmetry, 17(6), 844. https://doi.org/10.3390/sym17060844
39. Charilogis, V. & Tsoulos, I.G.(2022).Toward an Ideal Particle Swarm Optimizer for Multidimensional Functions. Information, 13, 217.Doi: https://doi.org/10.3390/info13050217
40. Lagaris, I.E. & Tsoulos, I.G. (2007). Stopping rules for box-constrained stochastic global optimization.Applied Mathematics and Computation 197 (2008) 622–632. Doi:10.1016/j.amc.2007.08.001
41. Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Chapters 3 & 5.
42. 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.
43. 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.
44. Gaviano, M., Ksasov, D. E., Lera, D., & Sergeyev, Y. D. (2003). Software for generation of classes of test functions with known local and global minima for global optimization. ACM Transactions on Mathematical Software, 29(4), 469–480.
45. Lennard-Jones, J. E. (1924). On the Determination of Molecular Fields. Proceedings of the Royal Society of London. Series A, 106(738), 463–477.
46. Zabinsky, Z. B., Graesser, D. L., Tuttle, M. E., & Kim, G. I. (1992). Global optimization of composite laminates using improving hit and run. In Recent Advances in Global Optimization (pp. 343–368).
47. 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.
48. Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation, 13(2), 398–417. Doi: https://doi.org/10.1109/TEVC.2008.927706
49. Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation, 10(6), 646–657. Doi: https://doi.org/10.1109/TEVC.2006.872133
50. Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation, 9(2), 159–195. Doi: https://doi.org/10.1162/106365601750190398
51. Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Transactions on Evolutionary Computation, 10(3), 281–295.Doi: https://doi.org/10.1109/TEVC.2005.857610
52. Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the american statistical association, 32(200), 675-701. Doi: https://doi.org/10.1080/01621459.1937.105035
53. Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. Heliyon, 4(11).
54. Suryadevara, S., & Yanamala, A. K. Y. (2021). A Comprehensive Overview of Artificial Neural Networks: Evolution, Architectures, and Applications. Revista de Inteligencia Artificial en Medicina, 12(1), 51-76.
55. Kelly, M., Longjohn, R., & Nottingham, K. (n.d.). The UCI Machine Learning Repository. Retrieved from https://archive.ics.uci.edu
56. Alcalá-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., García, S., Sánchez, L., & Herrera, F. (2011). KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing, 17, 255–287.
57. Weiss, S. M., & Kulikowski, C. A. (1991). Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems. Morgan Kaufmann Publishers Inc.
58. Tzimourta, K.D.; Tsoulos, I.; Bilero, I.T.; Tzallas, A.T.; Tsipouras, M.G.; Giannakeas, N. Direct Assessment of Alcohol Consumption in Mental State Using Brain Computer Interfaces and Grammatical Evolution. Inventions 2018, 3, 51.
59. Quinlan, J. R. (1987). Simplifying Decision Trees. International Journal of Man-Machine Studies, 27(3), 221–234.
60. Shultz, T., Mareschal, D., & Schmidt, W. (1994). Modeling Cognitive Development on Balance Scale Phenomena. Machine Learning, 16, 59–88.
61. Zhou, Z. H., & Jiang, Y. (2004). NeC4.5: neural ensemble based C4.5. IEEE Transactions on Knowledge and Data Engineering, 16(6), 770–773.
62. Setiono, R., & Leow, W. K. (2000). FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks. Applied Intelligence, 12(1), 15–25.
63. Demiroz, G., Govenir, H. A., & Ilter, N. (1998). Learning Differential Diagnosis of Eryhemato-Squamous Diseases using Voting Feature Intervals. Artificial Intelligence in Medicine, 13, 147–165.
64. P. Horton, K. Nakai, A Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins, In: Proceedings of International Conference on Intelligent Systems for Molecular Biology **4**, pp. 109-15, 1996.

65. Hayes-Roth, B., & Hayes-Roth, F. (1977). Concept learning and the recognition and classification of exemplars. Journal of Verbal Learning and Verbal Behavior, 16, 321–338.
66. Kononenko, I., Šimec, E., & Robnik-Šikonja, M. (1997). Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF. Applied Intelligence, 7, 39–55.
67. French, R. M., & Chater, N. (2002). Using noise to compute error surfaces in connectionist networks: a novel means of reducing catastrophic forgetting. Neural Computation, 14, 1755–1769.
68. Dy, J. G., & Brodley, C. E. (2004). Feature Selection for Unsupervised Learning. Journal of Machine Learning Research, 5, 845–889.
69. Perantonis, S. J., & Virvilis, V. (1999). Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis. Neural Processing Letters, 10, 243–252.
70. Garcke, J., & Griebel, M. (2002). Classification with sparse grids using simplicial basis functions. Intelligent Data Analysis, 6(5), 483–502.
71. McDermott, J., & Forsyth, R. S. (2016). Diagnosing a disorder in a classification benchmark. Pattern Recognition Letters, 73, 41–43.
72. Cestnik, G., Kononenko, I., & Bratko, I. (1987). Assistant-86: A Knowledge-Elicitation Tool for Sophisticated Users. In Bratko, I. & Lavrac, N. (Eds.), Progress in Machine Learning (pp. 31–45). Wilmslow: Sigma Press.
73. Elter, M., Schulz-Wendtland, R., & Wittenberg, T. (2007). The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process. Medical Physics, 34, 4164–4172.
74. M.A. Little, P.E. McSharry, S.J Roberts et al, Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection. BioMed Eng OnLine **6**, 23, 2007.
75. Little, M. A., McSharry, P. E., Roberts, S. J., Costello, D. A., & Moroz, I. M. (2007). Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection. BioMedical Engineering OnLine, 6(23). https://doi.org/10.1186/1475-925X-6-23
76. Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., & Johannes, R. S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care (pp. 261–265). IEEE Computer Society Press.
77. Lucas, D. D., Klein, R., Tannahill, J., Ivanova, D., Brandon, S., Domyancic, D., & Zhang, Y. (2013). Failure analysis of parameter-induced simulation crashes in climate models. Geoscientific Model Development, 6(4), 1157–1171.
78. Giannakeas, N., Tsipouras, M. G., Tzallas, A. T., Kyriakidi, K., Tsianou, Z. E., Manousou, P., Hall, A., Karvounis, E. C., Tsianos, V., & Tsianos, E. (2015). A clustering based method for collagen proportional area extraction in liver biopsy images. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS) (pp. 3097–3100).
79. Hastie, T., & Tibshirani, R. (1987). Non-parametric logistic and proportional odds regression. Journal of the Royal Statistical Society: Series C (Applied Statistics), 36(3), 260–276.
80. Dash, M., Liu, H., Scheuermann, P., & Tan, K. L. (2003). Fast hierarchical clustering and its validation. Data & Knowledge Engineering, 44, 109–138.
81. Gorman, R.P.; Sejnowski, T.J. Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets. Neural Netw. 1988, 1, 75–89.
82. Cortez, P., & Silva, A. M. G. (2008). Using data mining to predict secondary school student performance. In Proceedings of the 5th Future Business Technology Conference (FUBUTEC 2008) (pp. 5–12). EUROSIS-ETI.
83. Yeh, I.-C., Yang, K.-J., & Ting, T.-M. (2009). Knowledge discovery on RFM model using Bernoulli sequence. Expert Systems with Applications, 36(3), 5866–5871.
84. Jeyasingh, S., & Veluchamy, M. (2017). Modified bat algorithm for feature selection with the Wisconsin diagnosis breast cancer (WDBC) dataset. Asian Pacific journal of cancer prevention: APJCP, 18(5), 1257.
85. Alshayeji, M. H., Ellethy, H., & Gupta, R. (2022). Computer-aided detection of breast cancer on the Wisconsin dataset: An artificial neural networks approach. Biomedical signal processing and control, 71, 103141.
86. Raymer, M., Doom, T. E., Kuhn, L. A., & Punch, W. F. (2003). Knowledge discovery in medical and biological datasets using a hybrid Bayes classifier/evolutionary algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 33(5), 802–813.
87. Zhong, P., & Fukushima, M. (2007). Regularized nonsmooth Newton method for multi-class support vector machines. Optimization Methods and Software, 22(2), 225–236.
88. Andrzejak, R. G., Lehnertz, K., Mormann, F., Rieke, C., David, P., & Elger, C. E. (2001). Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: dependence on recording region and brain state. Physical Review E, 64(6), 061907.
89. Tzallas, A. T., Tsipouras, M. G., & Fotiadis, D. I. (2007). Automatic Seizure Detection Based on Time-Frequency Analysis and Artificial Neural Networks. Computational Intelligence and Neuroscience, 2007, Article ID 80510. https://doi.org/10.1155/2007/80510
90. M. Koivisto, K. Sood, Exact Bayesian Structure Discovery in Bayesian Networks, The Journal of Machine Learning Research **5**, pp. 549–573, 2004.
91. D. P. Kingma, J. L. Ba, ADAM: a method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), pp. 1–15, 2015.
92. K. O. Stanley, R. Miikkulainen, Evolving Neural Networks through Augmenting Topologies, Evolutionary Computation **10**, pp. 99-127, 2002.

93. Zhu, V., Lu, Y., & Li, Q. (2006). MW-OBS: An improved pruning method for topology design of neural networks. Tsinghua Science and Technology, 11(4), 307-312.
94. Grzegorz Klima, Fast Compressed Neural Networks, available from http://fcnn.sourceforge.net/.