Revista Facultad de Ingeniería (Rev. Fac. Ing.) Vol. 33, No. 69.
L-ISSN: 0121-1129, e-ISSN: 2357-5328.

DOI: 10.19053/01211129.v33.n69.2024.17895

# COMPARATIVE STUDY OF CUCKOO-INSPIRED ALGORITHMS TO SOLVE LARGE-SCALE CONTINUOUS OPTIMIZATION PROBLEMS

## Estudio comparativo de algoritmos inspirados en el cuco para problemas de optimización continua a gran escala

**Carlos-Alberto Cobos-Lozada**
PhD. Universidad del Cauca (Popayán-Cauca, Colombia).
ccobos@unicauca.edu.co

**Henry Muñoz-Collazos**
Ing, Credit One Bank (Las Vegas-Nevada, Estados, Unidos).
henry.munoz@creditone.com

**Richar Urbano-Muñoz**
Esp. Universidad del Cauca (Popayán-Cauca, Colombia).
rurbano@unicauca.edu.co

**Received / Recibido:** 13/04/2024
**Accepted / Aceptado:** 05/08/2024

## ABSTRACT
Two distinctive behaviors of the cuckoo bird have inspired several metaheuristic algorithms to solve continuous optimization problems. In addition to the well-known parasitic breeding behavior that gave rise to several cuckoo search (CS) algorithms, another behavior related to their clustering and the way they locate food sources has given rise to the COA algorithm. As a result, there are several variants to solve continuous optimization problems; however, it is necessary to define which one is the most suitable under specific requirements. This paper compares six of these algorithms, including CS+LEM (proposed in this paper), which consists of a hybridization of the CS algorithm with learning evolutionary models (LEM) using an approach known as "metaheuristics enhanced by artificial intelligence". Three assessments were performed using a set of 61 continuous test functions: 1) the optimal value achieved with a fixed execution time; 2) the number of objective function evaluations required to reach the global optimum; and 3) the optimal value achieved with a fixed number of objective function evaluations. CS+LEM presents the best results in evaluation 1, while COA presents the best results in evaluations 2 and 3. The results were analyzed using the Friedman and Wilcoxon nonparametric statistical tests.

**Keywords:** artificial intelligence; Cuckoo search algorithm; large-scale continuous problems; metaheuristics; optimization.

## RESUMEN
Dos comportamientos distintivos del pájaro cuco han inspirado varios algoritmos metaheurísticos para resolver problemas de optimización continua. Además del conocido comportamiento de reproducción parasitaria que dio origen a diversos algoritmos de búsqueda cuco (CS por sus siglas en inglés), otro comportamiento relacionado con sus agrupaciones y la forma en que localizan las fuentes de alimento ha dado lugar al algoritmo COA. Como resultado, existen diferentes variantes para resolver problemas de optimización continua; sin embargo, es necesario definir cuál es el más adecuado para resolver un

Esta edición se financió con recursos del Patrimonio Autónomo Fondo Nacional de Financiamiento para la Ciencia, la Tecnología y la Innovación, Francisco José de Caldas, Minciencias

problema bajo requerimientos específicos. En este trabajo se realiza una comparación entre seis de estos algoritmos incluido CS+LEM (propuesto en este artículo), una hibridación del algoritmo CS con modelos evolutivos que aprenden (LEM por sus siglas en inglés) usando un enfoque conocido como "metaheurística mejorada por inteligencia artificial". Se realizaron tres evaluaciones utilizando un conjunto de 61 funciones de prueba continuas: 1) el valor óptimo alcanzado con un tiempo fijo de ejecución; 2) el número de evaluaciones de la función objetivo necesarias para alcanzar el óptimo global; 3) el valor óptimo alcanzado con un número fijo de evaluaciones de la función objetivo. CS+LEM presenta los mejores resultados en la evaluación 1, mientras que COA presenta los mejores resultados en las evaluaciones 2 y 3. Los resultados se analizaron mediante las pruebas estadísticas no paramétricas de Friedman y Wilcoxon.

**Palabras clave:** algoritmo de búsqueda del cuco; inteligencia artificial; metaheurísticas; optimización; problemas continuos a gran escala.

## 1. INTRODUCTION

Metaheuristic algorithms are general-purpose algorithms used to find the solution to a specific problem. Metaheuristics are considered technical or high-level strategies that combine low-level techniques and tactics to explore and exploit the search space [1]. Among the best-known metaheuristics are Genetic Algorithm (GA) [2], Memetic algorithm [3], Tabu Search [4], Ant Colony Optimization [5], Particle Swarm Optimization (PSO) [6], Grey Wolf Optimization [7], Firefly algorithm [8]-[9], Differential Evolution (DE) [10], and Harmony Search algorithms [11]-[12].

In recent years, optimization algorithms have become one of the most essential fields in science and engineering —especially when minimizing time, costs, materials, and space—; therefore, this area is currently the object of scientific research and development, looking to improve productivity, cost, and processing time [13].

Cuckoo Search (CS) is a metaheuristic algorithm based on the exciting breeding behavior (brood parasitism) of certain species of cuckoo birds, in combination with the Lévy flight behavior of some birds or fruit flies [14]. The success of this algorithm versus Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) algorithms has made this useful for engineering optimization problems like the design of springs, design of beam structures, and data fusion in wireless sensor networks [14]-[15]. The advantage of CS against GA and PSO lies in the balance of randomization, intensification, and the lowest number of parameters to be controlled [14]. Cuckoo Optimization Algorithm (COA) is another metaheuristic algorithm inspired by cuckoo birds, but in this case, based on their lifestyle (immigration of societies or groups), that has shown promising results in continuous optimization problems. This paper compares six cuckoo-inspired algorithms to help solution designers select the appropriate algorithm for a specific optimization problem. Three scenarios of evaluation were used: 1) the optimal value achieved with a fixed time of execution, which is used for applications that need the best possible solution in a very short execution time (for online problems); 2) the number of objective function evaluations (OFE) required to reach the global optimum, it was designed to define the cuckoo-inspired algorithm that can solve (find optimal solutions) the greatest number of problems without restrictions on execution time (for offline optimization problems); and 3) the optimal value achieved with a fixed number of OFEs, where the behavior between online and offline applications based on the number of OFEs was analyzed; currently, this test is the least commonly used by the research community because algorithms can spend a variable time between each objective function evaluation (some algorithms spend excessive time, even greater than that required to evaluate the objective function, while others need much less time).

The paper is organized as follows: Section II summarizes five cuckoo-inspired algorithms; Section III presents a new algorithm proposed by the authors, in which CS is hybridized with Learnable Evolution Models (LEM); Section IV presents the analysis of the experimental results of the algorithm against a broad set of test functions; finally, some concluding remarks and suggestions for future work are presented.

## 2. CUCKOO-INSPIRED ALGORITHMS

Cuckoo Search has been applied to solve many engineering optimization problems and underwent several modifications that led to various improvements in accuracy and convergence time over general or specific optimization problems, e.g., decreasing the noise sensitivity of CS. The CS algorithm is described below, followed by three modifications that will be considered in the comparative study, namely: Improved Cuckoo Search Algorithm (ICS) [16], Modified Cuckoo Search (MCS) [17], and Modified Cuckoo Search Algorithm (MCSA) [18]. Finally, in this section, another algorithm inspired by the lifestyle of the cuckoo is presented, the Cuckoo Optimization Algorithm (COA) [19].

The **Cuckoo Search (CS)** algorithm provides a new way for intensification (search for better solutions in the neighborhood of the current solution) and diversification (make sure the algorithm can explore the search space efficiently). By simplifying the breeding behavior of the cuckoo, a set of three idealized rules can be established [14]: 1) Each cuckoo lays one egg at a time and deposits it in a randomly chosen nest; 2) The best nests with high-quality eggs will be carried over to the next generations; 3) The number of available host nests is fixed, and the host bird discovers the egg laid by a cuckoo with a probability $\rho_\alpha \in [0,1]$ ($\rho_\alpha$, Percentage of abandonments). In this case, the host bird can either get rid of the egg or abandon the nest and build an entirely new nest. Likewise, in nature, many animals and insects search for food through a random or quasi-random walk (since the next step is always based on the current location and the probability of moving to the following location) that can be modeled with a Lévy distribution (a continuous probability distribution for a non-negative random variable) known as Lévy flights [20]; many studies have shown that the flight behavior of some animals and insects follows its typical characteristics. This kind of search was included in the CS algorithm in Step 1, when the cuckoo randomly chose a nest.

The **Improved Cuckoo Search Algorithm (ICS)** was proposed by Valian, Mohanna, and Tavakoli [16]; the main difference with CS algorithm lies in the manner of adjusting the $\rho_\alpha$ and $\alpha$ parameters (the rate of abandonment and the Lévy flight step size, respectively). In ICS, the $\rho_\alpha$ and $\alpha$ values are dynamically changed with the number of generations. In the first generations, the values of $\rho_\alpha$ and $\alpha$ must be large enough to ensure that the algorithm increases the diversity of solution vectors (diversification). However, these values are decreased in the later generations to enable a better fit of the solution vectors (intensification).

The **Modified Cuckoo Search (MCS)** was proposed by Walton, Hassan, Morgan, and Brown [17]. It presents two modifications to the CS algorithm. The first change was made in the step size of Lévy flights ($\alpha$ parameter). The value of $\alpha$ decreases as the number of generations increases, thus increasing the intensification. The second modification was to add an information exchange between eggs to accelerate the convergence to the optimal solution. In the CS algorithm, there is no exchange of information between individuals, and searches are performed independently. In this version, a fraction of the eggs with the best fitness is in a “top” group of eggs. For each of these, an egg is randomly selected from the top eggs, and a new egg is generated on the line connecting the two eggs. The new egg is generated at the midpoint of both eggs using the Golden Ratio (an irrational mathematical constant frequently present in distance ratios taken of the simple geometric figures such as pentagon, pentagram, decagon, and dodecahedron, and defined by $\varphi=(1+\sqrt{5}/2)$). A significantly better performance was achieved using Golden Radio than with a random fraction.

The **Modified Cuckoo Search Algorithm (MCSA)** proposed by Tuba, Subotiv, and Stanarevic [18] proposed a change in how to calculate the size of the random steps. The change includes a function that allows sorting the matrix of candidate solutions (nests) for the fitness value of the solutions contained. Thus, the solutions with higher fitness have a slight advantage over those with lower fitness. This method maintains the selection pressure (the degree to which high-fitness level solutions are selected) toward the best solutions, thus facilitating better results.

The **Cuckoo Optimization Algorithm (COA)** is inspired by the lifestyle of cuckoo birds and was proposed in 2011 by Rajabioun [19]. In COA, a habitat (a matrix of size $N_{pop}$ x $M_{dim}$) is generated with random points, and the utility function is calculated for each. A specific number of eggs (5-20) and an Egg Laying Radius (ELR) that defines the maximum distance where cuckoos can host their eggs are assigned to each cuckoo. In each generation, zones (groups) are defined for the habitat of each cuckoo. Each area is evaluated and a utility value is assigned to it; this value represents the survival rate for an individual and its eggs, being defined by the number of individuals that have proliferated in the area, the availability of food, and the similarities between the characteristics of the cuckoo eggs and the host bird eggs. Accordingly, the area with the highest utility value is set as the target point, that is, the best migration habitat for mature cuckoos who establish a flight path defined by a percentage of the distance $\lambda$, and a deflection angle $\phi$ ($-\pi/6$, $\pi/6$ radians). Generation after generation, the cuckoos are unevenly distributed over the search space, and thus it is complex to identify the cuckoos that belong to each group; therefore, the k-means clustering algorithm is used (where k is between 3 and 5) to help in adequately defining the cuckoos in each group.

## 3. PROPOSED CS+LEM ALGORITHM

Inspired by the concept of Learnable Evolution Models (LEM) proposed by Michalski [21][22], a new version of the Cuckoo Search (CS) algorithm is proposed in this section. In LEM, machine learning techniques are used to generate new populations along with the Darwinian method, applied in evolutionary computation and based on mutation and natural selection. This method can determine which individuals in a population (or set of individuals from previous populations) are better than others in performing specific tasks. This reasoning, expressed as an inductive hypothesis, is used to generate new populations. Then, when the algorithm is running in Darwinian evolution mode, it uses random or semi-random operations to generate new individuals (employing traditional mutation or recombination techniques).

In this research, the algorithm proposed in [23] carries out the machine learning process, which is responsible for the rule inference process. The latter defines a set of conjunctive rules ($P\leftarrow R_1\wedge R_2\wedge\ldots\wedge R_n$) that delineate the regions where there is a greater chance of finding a better value for each dimension $x_i$ (for example $LV_{xi}\leq x_i\leq HV_{xi}$; here, LV and HV are the lower and upper limits of the rules for the value on dimension). Given the combination of rules (R) for each dimension, the search space is limited to regions most likely to generate a global optimum. The rule inference process is run for the first time immediately after creating the initial population of nests. The steps of the rule inference process are summarized in Figure 1.

The new proposal is called Cuckoo Search using Learnable Evolution Models (CS+LEM), and the steps of the algorithm are presented in Figure 2. If the LEM process is activated (when then LEM variable is set to true), every time the population of the nest changes, the rule inference process is executed to update rules for each dimension. If the LEM process is activated, the new cuckoo randomly generated via Lévy flights is mutated in some dimensions. The process mutates a dimension with a probability defined by the rule consideration rate (RCR) parameter. By default, the RCR parameter is set to 0.5, meaning half of the dimensions are created via Lévy flights, and the other half are created based on rules. When the algorithm creates a specific number of cuckoos without managing to improve the fitness function (MNIWI parameter), the LEM process is deactivated. Then if the algorithm repeats this situation without improving fitness for a specific MNIWI, the LEM process is activated again, and so on.

```
begin
1. From the current population (CP) of nests, the high-performance group and low performance group are chosen using
   the following formula: Hgroup = CP (1..., ⌊n/2⌋), Lgroup = CP (n-⌊n/2⌋,...,n), where n is the number of host nest and ⌊n/2⌋
   is the size of high performance and low performance groups.
2. A matrix is generated for each class {high, low} belonging to each group. This matrix stores the attribute value in that
   position and the corresponding fitness as follows, if it is Hgroup it is assigned 1, otherwise 0 is assigned. It is then put in
   order according to the value of its attribute, from lowest to highest.
3. The probability of occurrence of each attribute is calculated.
4. The attribute with the highest probability of occurrence is selected and added to the list of rules P.
5. Steps 2, 3 and 4 are repeated until all the attributes of each group have been evaluated. The resulting rule is of type P →
   Q, where P is a conjunction of the rules that have the highest probability for each attribute, and Q corresponds to class
   1 (high-performance group).
end
```

**Figure 1.** *Rule inference process*

## 4. EXPERIMENTATION

This section shows the performance of cuckoo-inspired algorithms in three different assessments, namely: 1) the optimal value achieved with a fixed time of execution; 2) the number of objective function evaluations required to reach the global optimum; and 3) the optimal value achieved with a fixed number of Objective Function Evaluations (OFE). Previously used parameters for executing all algorithms in each experiment are presented in Table 1. The parameters for all algorithms were used based on values recommended by their original authors. The approach of different challenges in optimization problems was used in our work. Therefore, the parameter values were the same for all optimization problems, i.e., there was no tuning of parameters conducted for each algorithm in each specific problem.

Table 2 shows 6 unimodal separable functions, 22 unimodal non-separable functions, 7 multimodal separable functions, and 26 multimodal non-separable functions. They are based on those proposed in the report “Benchmark Functions for the CEC’2010 Special Session and Competition on Large-Scale Global Optimization” [24][25][26] and the paper “A comparative study of Artificial Bee Colony Algorithm” [9], which provides an adequate range of complexity levels. For each function, the global minimum is searched.

For each assessment, the average and standard deviation of 30 independent executions were reported per function. The initial population is generated randomly within ranges specified for each function. The results were obtained using the same computer configuration (hardware and software). All algorithms were implemented on C# language programming.

```
begin
    Objective function F = f(x); x = $(x_1, x_2..., x_d)^T$
    Generate initial population of n host nests xi (i=1, 2, ..., n)
    LEM = true
    iterationsWithoutImprovement = 0
    while (t <MaxGeneration) or (stop criterion) do
        Get a new cuckoo randomly by Lévy flights (say, s)
        Foreach dimension dim of the new cuckoo s
            If (LEM)
                If (U(0,1) < RCR)
                    s[dim] = $U(LB^{best}_{dim}, UB^{best}_{dim})$, where best is the best set of rules for current dimension
                end if
            end if
        End foreach
        Evaluate the Quality(Fitness) of s, Fs
        Choose randomly a nest among n (say, j)
        If ($F_s$ is better than $F_j$) // < for minimizing and > for maximizing
            Replace j by the new solution s
            If (LEM) Run the rule inference process
        Else
            iterationsWithoutImprovement ++
            If (iterationsWithoutImprovement > MNIWI)
                iterationsWithoutImprovement = 0
                LEM = not LEM
            end if
        end if
        A fraction ($\rho_\alpha$) of worse nests is abandoned and new ones are built
        Keep the best solutions (or nests with quality solutions)
        Rank the solutions and find the current best
    end while
    Postprocess results and visualization
end
```

**Figure 2.** *Pseudo-code of CS+LEM*

**Table 1**. General Settings for Experiments

| Variable/Parameter | CS | COA | ICS | MCS | MCSA | CS+LEM |
|---|---|---|---|---|---|---|
| Number of nests (Nn) | 30 | 30 | 30 | 30 | 30 | 30 |
| Number of dimensions (Nd) | 30 | 30 | 30 | 30 | 30 | 30 |
| Percentages of abandonments (Pa) | 0.25 | N.A. | 0.25 | 0.75 | 0.25 | 0.25 |
| Maximum number of cuckoos (Mc) | N.A. | 50 | N.A. | N.A. | N.A. | N.A. |
| Number of clusters (k) | N.A. | 2 | N.A. | N.A. | N.A. | N.A. |
| Maximum number of objective function evaluations (Me) | N.A. | N.A. | 50,000 | N.A. | N.A. | N.A. |
| Maximum number of steps (Ms) | N.A. | N.A. | N.A. | 100 | N.A. | N.A. |
| Maximum Lévy step size (Ml) | N.A. | N.A. | N.A. | 0.01 | N.A. | N.A. |
| Rule Consideration Rate (RCR) | N.A. | N.A. | N.A. | N.A. | N.A. | 0.5 |
| Maximum Number of Iterations Without Improvements (MNIWI) | N.A. | N.A. | N.A. | N.A. | N.A. | 120 |
| Bandwidth (Bw) | N.A. | N.A. | N.A. | N.A. | N.A. | 0.001 |

**Table 2.** Benchmark Functions (D: Dimensions, C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable).

| No | Function | Range | D | C |
|---|---|---|---|---|
| 1 | StepInt [9] | [-5.12, 5.12] | 5 | U,S |
| 2 | Step [18] | [-100, 100] | 30 | U,S |
| 3 | Sphere (De Jong's First Function [16] [26]) | [-100,100] | 30 | U,S |
| 4 | Sum of Squares [9] | [-10,10] | 30 | U,S |
| 5 | Quartic (De Jong's Forth Function) [9] | [-1.28, 1.28] | 30 | U,S |
| 6 | Sum of Different Power [27] | [-10, 10] | 30 | U,S |
| 7 | Beale [9] | [-4.5, 4.5] | 2 | U,N |
| 8 | Easom [26] | [-100, 100] | 2 | U,N |
| 9 | Matyas [9] | [-10, 10] | 2 | U,N |
| 10 | Colville [9] | [-10, 10] | 4 | U,N |
| 11 | Zakharov [9] | [-5, 10] | 30 | U,N |
| 12 | Schwefel's Problem 2.22 [9] | [-10, 10] | 30 | U,N |
| 13 | Schwefel's Problem 1.2 [9] [26] | [-10, 10] | 30 | U,N |
| 14 | Rosenbrock [26] | [-100, 100] | 30 | U,N |
| 15 | Dixon-Price [18] | [-10, 10] | 30 | U,N |
| 16 | Single-group Shifted and m-rotated Elliptic [26] | [-100, 100] | 30 | U,N |
| 17 | Single-group Shifted m-dimensional Schwefel's Problem 1.2 [26] | [-100, 100] | 30 | U,N |
| 18 | D/2m-group Shifted and m-rotated Elliptic [9] | [-100, 100] | 30 | U,N |
| 19 | D/2m-group Shifted m-dimensional Schwefel's Problem 1.2 [26] | [-100, 100] | 30 | U,N |
| 20 | D/m-group Shifted and m-rotated Elliptic [26] | [-100, 100] | 30 | U,N |
| 21 | D/m-group Shifted m-dimensional Schwefel's Problem 1.2 [26] | [-100, 100] | 30 | U,N |
| 22 | Shifted Schwefel's Problem 1.2 [26] | [-100, 100] | 30 | U,N |
| 23 | Shifted Elliptic [26] | [-100, 100] | 30 | U,N |
| 24 | Shifted Schwefel's Problem 1.2 with Noise in Fitness [24] | [-100, 100] | 30 | U,N |
| 25 | Shifted Rotated High Conditioned Elliptic [24] | [-100, 100] | 30 | U,N |
| 26 | Elliptic [26] | [-100, 100] | 30 | U,N |
| 27 | Trid 10 [9] |  | 10 | U,N |
| 28 | Powell [9] | [-4, 5] | 24 | U,N |
| 29 | Bohachevsky 1 [9] | [-100, 100] | 2 | M,S |
| 30 | Booth [9] | [-10, 10] | 2 | M,S |
| 31 | Rastrigin [26] | [-5.12, 5.12] | 30 | M,S |
| 32 | Generalized Schwefel [26] | [-500, 500] | 30 | M,S |
| 33 | Michalewicz 10 [26] | [0, ] | 30 | M,S |
| 34 | Shifted Rastrigin [26] | [-5, 5] | 30 | M,S |
| 35 | Shifted Ackley [26] | [-32, 32] | 30 | M,S |
| 36 | Schaffer [9] | [-100, 100] | 2 | M,N |
| 37 | Six-Hump Camel-Back [9] | [-5, 5] | 2 | M,N |
| 38 | Bohachevsky 2 [9] | [-100, 100] | 2 | M,N |

**Table 2.** Benchmark Functions (D: Dimensions, C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable) (Continued).

| | | | | |
|---|---|---|---|---|
| 39 | Bohachevsky 3 [9] | [-100, 100] | 2 | M,N |
| 40 | Shubert [26] | [-10, 10] | 5 | M,N |
| 41 | Goldstein & Price [27] | [-2, 2] | 2 | M,N |
| 42 | Griewank [26] | [-600, 600] | 30 | M,N |
| 43 | Ackley [26] | [-32, 32] | 30 | M,N |
| 44 | Single-group Shifted and m-rotated Rastrigin [26] | [-5, 5] | 30 | M,N |
| 45 | Single-group Shifted and m-rotated Ackley [26] | [-32, 32] | 30 | M,N |
| 46 | Single-group Shifted m-dimensional Rosenbrock [26] | [-100, 100] | 30 | M,N |
| 47 | D/2m-group Shifted and m-rotated Rastrigin [26] | [-5, 5] | 30 | M,N |
| 48 | D/2m-group Shifted and m-rotated Ackley [26] | [-32, 32] | 30 | M,N |
| 49 | D/2m-group Shifted m-dimensional Rosenbrock [26] | [-100, 100] | 30 | M,N |
| 50 | D/m-group Shifted and m-rotated Rastrigin [26] | [-5, 5] | 30 | M,N |
| 51 | D/m-group Shifted and m-rotated Ackley [26] | [-32, 32] | 30 | M,N |
| 52 | D/m-group Shifted m-dimensional Rosenbrock [26] | [-100, 100] | 30 | M,N |
| 53 | Shifted Rosenbrock [26] | [-100, 100] | 30 | M,N |
| 54 | Shifted Rotated Expanded Scaffer's F6 [24] | [-100, 100] | 30 | M,N |
| 55 | Shifted Rotated Weierstrass [24] | [-0.5, 0.5] | 30 | M,N |
| 56 | Shekel's foxholes [27] | [-65.536, 65.536] | 4 | M,N |
| 57 | Kowalik [9] | [-5, 5] | 4 | M,N |
| 58 | Perm [9] | [-, ] | 4 | M,N |
| 59 | Power Sum [9] | [0, ] | 4 | M,N |
| 60 | Hartman 6 [9] | [0, 1] | 6 | M,N |
| 61 | Langerman 5 [27] | [0, 10] | 5 | M,N |

### *A. Assessment 1: Best Optimal Value Reached at Different Times*

This assessment aims to identify which cuckoo-inspired algorithm provides the best results (results nearest to the global minimum of the objective function) for the following periods: 5, 10, 20, 40, and 80 seconds. This section shows results in general (all functions as a whole). Additional analyses over four groups of functions (unimodal separable, unimodal non-separable, multimodal separable, and multimodal non-separable) were performed; however, the results are not presented in this paper due to space limitations.

The Friedman test shows that CS+LEM is the best option for solving problems when the designer does not know anything about the landscape of the fitness function, the problem has high dimensionality, and the execution time is short (lower than 80 seconds) (see Table 3). General results based on the Wilcoxon non-parametric test with a 0.95 significance level also show that: 1) At 5s, CS+LEM outperforms COA, MCS, and MCSA, and ICS and MCSA outperform COA and MCS; 2) At 10s and 20s, CS+LEM outperforms all other algorithms, and MCSA outperforms CS, COA, ICS, and MCS; 3) At 40s and 80s, CS+LEM outperforms all other algorithms and COA outperforms CS, ICS, MCS and MCSA.

**Table 3.** Ground-Truth Friedman Test Rankings for Assessment Related to Time (distributed according to chi-square with 5 degrees of freedom)

| Algorithm | 5 seconds | | 10 seconds | | 20 seconds | | 40 seconds | | 80 seconds | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Ranking | Position | Ranking | Position | Ranking | Position | Ranking | Position | Ranking | Position |
| CS | 3.4508 | 4 | 3.2951 | 3 | 3.3689 | 3 | 3.4508 | 4 | 3.4508 | 3 |
| COA | 3.6230 | 6 | 3.7623 | 5 | 3.6803 | 5 | 3.4098 | 3 | 3.4918 | 5 |
| ICS | 3.4508 | 3 | 3.4918 | 4 | 3.4590 | 4 | 3.4016 | 2 | 3.4180 | 4 |
| MCS | 3.5738 | 5 | 4.1557 | 6 | 4.1721 | 6 | 4.1967 | 6 | 4.2049 | 6 |
| MCSA | 3.4508 | 2 | 3.2459 | 2 | 3.2869 | 2 | 3.4590 | 5 | 3.3852 | 2 |
| CS+LEM | 3.4508 | 1 | 3.0492 | 1 | 3.0328 | 1 | 3.0820 | 1 | 3.0492 | 1 |
| Friedman statistic | 0.526932 | | 14.093677 | | 13.36534 | | 11.887588 | | 12.592506 | |
| p-value | 0.9911039254 | | 0.0150251913 | | 0.0201853957 | | 0.0363609785 | | 0.0275118689 | |

### *B. Assessment 2: Number of Objective Function Evaluations (OFE) Required to Reach the Global Optimum*

This assessment aims to identify which cuckoo-inspired algorithm provides the best results (results nearest to the global minimum of the objective function with the lowest number of evaluations). All algorithms were executed until the global minimum was found or a maximum of 50000 OFEs was exceeded.

On average, COA reports better results on the number of OFEs required to reach the global optimum than the other five algorithms supported by the Friedman non-parametric test (see Table 4). The main problem with COA is the additional execution time required by each iteration to cluster cuckoos (solutions) using the k-means algorithm because clustering in a continuous space is an NP-hard problem. The k-means algorithm requires O (n*k*t) operations to converge a non-optimal cluster solution, where n is the number of cuckoos, k is the number of clusters, and t is the average cycles (iterations) required by the algorithm to converge. Therefore, COA is more than six times slower than MCS (643.91%), which reports the lowest average time for each generation.

Additionally, COA outperforms all algorithms; CS outperforms ICS and MCS with a significance level=0.95, and CS+LEM outperforms MCS with a significance level=0.95 in Wilcoxon test results. It is essential to highlight that COA is the best cuckoo-inspired algorithm in this test because it reaches the optimum value in 55 test functions out of 61 functions in the maximum specified number of OFEs; this is a high number of resolved test functions in comparison with other cuckoo-inspired algorithms which can solve only 7 to 14 test functions using 50000 OFEs. It can be further noted that CS+LEM ranks second with 14 resolved test functions. Based on the above, COA is the best alternative for environments where much time can be expected to find the optimal solution.

### *C. Assessment 3: Best Optimal Value Reached at Different Number of Objective Function Evaluations*

This assessment aims to identify which cuckoo-inspired algorithm provides the best results (the results nearest to a global minimum for the objective function) at 5000, 10000, 20000, and 50000 OFEs. Friedman analysis shows that COA reports the best results for the highest number of OFEs, while MCS is best for a small number (Table 5).

**Table 4.** Ground-Truth Friedman Test Rankings for Assessment Related to Number of OFEs (distributed according to chi-square with 5 degrees of freedom)

| Algorithm | Ranking | Position |
|---|---|---|
| CS | 3.7951 | 3 |
| COA | 1.3279 | 1 |
| ICS | 4.0492 | 5 |
| MCS | 4.2787 | 6 |
| MCSA | 3.8197 | 4 |
| CS+LEM | 3.7295 | 2 |
| Friedman statistic | 102.271663 | |
| p-value | 8.50330E-11 | |

**Table 5.** Ground-Truth Friedman Test Rankings for Assessment Related to Best Optimal Value (distributed according to chi-square with 5 degrees of freedom)

| Algorithm | 5000 evaluations | | 10000 evaluations | | 20000 evaluations | | 50000 evaluations | |
|---|---|---|---|---|---|---|---|---|
| | Ranking | Position | Ranking | Position | Ranking | Position | Ranking | Position |
| CS | 3.2623 | 3 | 3.3443 | 3 | 3.4016 | 3 | 3.4918 | 3 |
| COA | 4.4016 | 6 | 3.6639 | 5 | 3.0574 | 1 | 2.9262 | 1 |
| ICS | 3.0410 | 2 | 3.1475 | 2 | 3.2131 | 2 | 2.9426 | 2 |
| MCS | 2.9180 | 1 | 3.1311 | 1 | 3.4098 | 4 | 3.5820 | 5 |
| MCSA | 3.3361 | 4 | 3.4016 | 4 | 3.4754 | 5 | 3.5656 | 4 |
| CS+LEM | 4.0410 | 5 | 4.3115 | 6 | 4.4426 | 6 | 4.4918 | 6 |
| Friedman statistic | 30.297424 | | 17.0726 | | 20.655738 | | 28.489461 | |
| p-value | 1.28878E-5 | | 0.0043641482 | | 9.407719E-4 | | 2.91959E-5 | |

Results using Wilcoxon test (by default with a significance level of 0.95) show that at 5000 OFEs, MCS outperforms CS, COA, ICS, and CS+LEM, and ICS outperforms COA and CS+LEM; at 10000 OFEs, MCS outperforms CS, COA, ICS, and CS+LEM, and ICS outperforms COA and CS+LEM; At 20000 OFEs, COA and ICS outperform CS and CS+LEM, and COA outperforms ICS, MCSA, and CS+LEM [Wilcoxon significance level of 0.90]; at 50000 OFE - COA outperforms CS, CSA, and CS+LEM, and ICS outperforms CS and CS+LEM. It is essential to consider that this test is the least used by the research community since algorithms that spend much time calculating the changes in the current individuals to obtain the new ones have a clear advantage over those that spend less time.

## 5. CONCLUSIONS AND FUTURE WORK

This paper presents a new CS algorithm called Cuckoo Search using Learnable Evolution Models, or CS+LEM. The proposed algorithm uses LEM techniques to create rules that enable inferring new candidates in the population that emerges not only from the random scan. The algorithm was subjected to 61 classic optimization features and obtained the best results in most of the functions when the designer does not know anything about the landscape of the fitness function, the problem has high dimensionality, and the execution time is short (Assessment 1).

The hybridization of CS with K-means (COA) reports better results in Test 2 (number of objective function evaluations required to reach the global optimum) than the other five algorithms. It is essential to highlight that COA reaches the optimum value for 55 test functions over 61 total functions; this is a high number of resolved test functions compared with other cuckoo-inspired algorithms that can only solve 7 to 14 test functions using 50000 OFEs. Unfortunately, COA requires a much longer execution time than the other CS algorithms. Therefore, COA is the best option for offline scenarios where users can wait much more time for the global optimum solution of the problem. Additionally, the CS+LEM algorithm ranks second in this assessment (Assessment 2).

COA and MCS report the best results in Assessment 3 (Best optimal value reached at different number of OFEs); for a high number of OFEs, COA reports the best results, while the MCS Algorithm reports the best results for a small number of evaluations.

The different methods for diversification and intensification employed by the cuckoo-inspired algorithms compared in the paper mean that each algorithm can provide a better solution to a specific problem in a specific scenario. This concept is supported by the "no-free lunch theorem for optimization" [28]. The paper helps define the best algorithms for three scenarios and allows designers to select the most appropriate algorithm for a specific optimization problem. Regarding future work, the research group proposes to compare the previously studied cuckoo-inspired algorithms exhaustively and in detail with other heuristics like PSO, DE, HS, and ABC.

## AUTHORS' CONTRIBUTION

**Carlos-Alberto Cobos-Lozada**: Conceptualization; Methodology; Formal Analysis; Investigation; Supervision; Writing-review and editing. **Henry Muñoz-Collazos**: Software; Formal Analysis; Investigation; Validation; Writing-original draft. **Richar Urbano-Muñoz**: Software; Formal Analysis; Investigation; Validation; Writing-original draft.

## ACKNOWLEDGMENTS

The University of Cauca partially supported this work.

## REFERENCES

[1] L. Velasco, H. Guerrero, and A. Hospitaler, "A Literature Review and Critical Analysis of Metaheuristics Recently Developed," *Arch. Comput. Methods Eng.*, vol. 31, 2023. https://doi.org/10.1007/s11831-023-09975-0

[2] R. R. Abo-Alsabeh and A. Salhi, "The Genetic Algorithm: A study survey," *Iraqi J. Sci.*, vol. 63, no. 3, pp. 1215-1231, 2022, https://doi.org/10.24996/ijs.2022.63.3.27

[3] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review," *Swarm Evol. Comput.*, vol. 2, pp. 1-14, 2012. https://doi.org/10.1016/j.swevo.2011.11.003.

[4] V. K. Prajapati, M. Jain, and L. Chouhan, "Tabu Search Algorithm (TSA): A Comprehensive Survey," in *Proceedings of 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things, ICETCE 2020*, 2020, pp. 222–229. https://doi.org/10.1109/ICETCE48199.2020.9091743

[5] N. Nayar, S. Gautam, P. Singh, and G. Mehta, "Ant Colony Optimization: A Review of Literature and Application in Feature Selection," in *Lecture Notes in Networks and Systems*, 2021, pp. 285-297. https://doi.org/10.1007/978-981-33-4305-4_22

[6] A. G. Gad, "Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review," *Arch. Comput. Methods Eng.*, vol. 29, no. 5, pp. 2531-2561, 2022. https://doi.org/10.1007/s11831-021-09694-4

[7] I. Sharma, V. Kumar, and S. Sharma, "A Comprehensive Survey on Grey Wolf Optimization," *Recent Adv. Comput. Sci. Commun.*, vol. 15, no. 3, pp. 323–333, 2022. https://doi.org/10.2174/2666255813999201007165454

[8] J. Li, X. Wei, B. Li, and Z. Zeng, "A survey on firefly algorithms," *Neurocomputing*, vol. 500, pp. 662-678, 2022. https://doi.org/10.1016/j.neucom.2022.05.100

[9] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," *Appl. Math. Comput.*, vol. 214, no. 1, pp. 108-132, 2009. https://doi.org/10.1016/j.amc.2009.03.090

[10] M. F. Ahmad, N. A. M. Isa, W. H. Lim, and K. M. Ang, "Differential evolution: A recent review based on state-of-the-art works," *Alexandria Eng. J.*, vol. 61, no. 5, pp. 3831-3872, 2022. https://doi.org/10.1016/j.aej.2021.09.013

[11] F. Qin, A. M. Zain, and K.-Q. Zhou, "Harmony search algorithm and related variants: A systematic review," *Swarm Evol. Comput.*, vol. 74, 2022. https://doi.org/10.1016/j.swevo.2022.101126

[12] E. Ruano-Daza, C. Cobos, J. Torres-Jimenez, M. Mendoza, and A. Paz, "A multiobjective bilevel approach based on global-best harmony search for defining optimal routes and frequencies for bus rapid transit systems," *Appl. Soft Comput. J.*, vol. 67, pp. 567-583, Jun. 2018. https://doi.org/10.1016/j.asoc.2018.03.026

[13] S. Salhi and J. Thompson, "An overview of heuristics and metaheuristics," in *The Palgrave Handbook of Operations Research*, 2022, pp. 353-403. https://doi.org/10.1007/978-3-030-96935-6_11

[14] X.-S. Yang and S. Deb, "Cuckoo Search via Lévy flights," in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 2009, pp. 210-214. https://doi.org/10.1109/NABIC.2009.5393690.

[15] K. Safdar, K. N. Abdul Rani, H. A. Rahim, S. J. Rosli, and M. A. Jamlos, "A Review on Research Trends in using Cuckoo Search Algorithm: Applications and Open Research Challenges," *Prz. Elektrotechniczny*, vol. 1, no. 5, pp. 18-24, 2023. https://doi.org/10.15199/48.2023.05.04.

[16] E. Valian, S. Mohanna, and S. Tavakoli, "Improved Cuckoo Search Algorithm for Feed forward Neural Network Training," *Int. J. Artif. Intell. Appl.*, vol. 2, no. 3, pp. 36-43, 2011. https://doi.org/10.5121/ijaia.2011.2304.

[17] S. Walton, O. Hassan, K. Morgan, and M. R. Brown, "Modified cuckoo search: A new gradient free optimisation algorithm," *Chaos, Solitons and Fractals*, vol. 44, no. 9, pp. 710-718, 2011. https://doi.org/101016/jchaos201106004

[18] M. Tuba, M. Subotic, and N. Stanarevic, "Modified Cuckoo Search Algorithm for Unconstrained Optimization Problems," *Proc. 5th Eur. Conf. Eur. Comput. Conf.*, pp. 263-268, 2011, [Online]. Available: http://www.wseas.us/e-library/conferences/2011/Paris/ECC/ECC-43.pdf

[19] R. Rajabioun, "Cuckoo Optimization Algorithm," *Appl. Soft Comput.*, vol. 11, no. 8, pp. 5508-5518, 2011. https://doi.org/doi.org/10.1016/j.asoc.2011.05.008

[20] J. Li, Q. An, H. Lei, Q. Deng, and G.-G. Wang, "Survey of Lévy Flight-Based Metaheuristics for Optimization," *Mathematics*, vol. 10, no. 15, 2022. https://doi.org/10.3390/math10152785

[21] R. S. Michalski, "Learnable evolution model: evolutionary processes guided by machine learning," *Mach. Learn.*, vol. 38, no. 1, pp. 9-40, 2000. https://doi.org/10.1023/a:1007677805582

[22] A. L. da Costa Oliveira, A. Britto, and R. Gusmão, "Machine learning enhancing metaheuristics: a systematic review," *Soft Comput.*, 2023. https://doi.org/10.1007/s00500-023-08886-3

[23] C. Cobos, D. Estupiñán, and J. Pérez, "GHS + LEM: Global-best Harmony Search using learnable evolution models," *Appl. Math. Comput.*, vol. 218, no. 6, pp. 2558-2578, 2011. https://doi.org/10.1016/j.amc.2011.07.073.

[24] P. N. Suganthan *et al.*, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Singapore, 2005. [Online]. Available: http://www.cmap.polytechnique.fr/~nikolaus.hansen/Tech-Report-May-30-05.pdf

[25] K. Tang *et al.*, "Benchmark functions for the CEC'2008 special session and competition on large scale global optimization," in *IEEE World Congress on Computational Intelligence*, Rio de Janeiro, Brazil: IEEE, 2008, pp. 1-18. [Online]. Available: http://sci2s.ugr.es/programacion/workshop/Tech.Report.CEC2008.LSGO.pdf

[26] T. Ke, L. Xiaodong, S. P. N., Y. Zhenyu, and W. Thomas, "Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization," Shanghai, China, 2010. [Online]. Available: http://goanna.cs.rmit.edu.au/~xiaodong/cec13-lsgo/competition/cec2013-lsgo-benchmark-tech-report.pdf

[27] M. Molga and C. Smutnicki, "Test functions for optimization needs," *Test functions for optimization needs*, no. c. pp. 1-43, 2005. [Online]. Available: https://robertmarks.org/Classes/ENGR5358/Papers/functions.pdf

[28] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67-82, 1997. https://doi.org/10.1109/4235.585893.