International Journal of Mathematical, Engineering and Management Sciences
Vol. 11, No. 1, 332-352, 2026
https://doi.org/10.33889/IJMEMS.2026.11.1.014

Adaptive Population Learning EJaya Algorithm for Real-World Optimization Problems

Sonal Deshwal
Department of Mathematics, University Institute of Sciences,
Chandigarh University, Mohali, Punjab, India.
E-mail: sonaldeswal1995@gmail.com

Sandeep Kumar Mogha
Department of Mathematics,
University of Delhi, Delhi, India.
&
Department of Mathematics, University Institute of Sciences,
Chandigarh University, Mohali, Punjab, India.
Corresponding author: moghadma@gmail.com, smogha@maths.du.ac.in

(Received on October 19, 2025; Revised on November 11, 2025; Accepted on December 2, 2025)

Abstract
EJaya enhances the global exploration capability of the Jaya algorithm but still suffers from drawbacks such as local stagnation and slow convergence due to its single learning strategy and weak population maintenance. To address these issues, this paper proposes an adaptive population-learning based improved variant, termed "PLEJaya." In the PLEJaya algorithm, first an adaptive learning method improves the EJaya performance by refining the initial population and then a linear reduction method diminishes the worthless members from the population and thus improves the overall algorithm's performance. The performance of PLEJaya has been examined on 53 benchmark functions, including 23 standard and 30 CEC 2017 test suite's benchmark functions, and compared with eight established meta-heuristic algorithms such as TLBO, EJAYA, JAYA, PSO, GA, AOA, GWO, and WOA. Additionally, the practicality of PLEJaya has also been confirmed on 4 constrained engineering design applications. The experimental results confirm that the proposed PLEJaya solution significantly outperforms its competitors in terms of accuracy as well as convergence rate and thus provides a viable alternative to current optimization techniques.

Keywords- Optimization, Meta-heuristics, Engineering optimization, Jaya, EJaya, Constrained.

1. Introduction

Optimization involves finding the optimal value of decision variables to address a specific problem, a concept widely applicable across diverse fields and numerous applications (Talbi, 2009). There are a lot of tough optimization problems in the real world where it is very hard to find the global optimal answer. Because of this, many algorithms have been created to deal with these kinds of problems. These algorithms can be broken down into two categories: exact and approximate approaches.

Exact methods guarantee the best solution within a reasonable timeframe, except in cases classified as NP-Hard problems where achieving a polynomial time solution is infeasible, resulting in significant computational demands. As a result, approximate methods have gotten more attention over the last 30 years, with the main goal of finding best solutions within acceptable timeframes.

A novel class of approximate algorithms, referred to as metaheuristics (Glover, 1986), has achieved importance in recent periods due to their simplicity, ease of implementation, and capability to navigate away from local optima, making them particularly suited for derivative-free problems. The two most

defining features of metaheuristics are their ability to exploit and explore. An algorithm's exploration capability is its capacity to find new search regions. The process of discovering the optimal solution in promising regions of the solution space is called exploitation. An effective metaheuristic strikes a balance between exploring and exploiting optimal results.

A vast variety of metaheuristics may be broken down into four main types, which are as follows:

i) *Swarm intelligence techniques*: The algorithms used in Swarm Intelligence are based on principles found in nature, namely in the behaviour of social insects and animals. Swarm intelligence is a type of artificial intelligence whereby each individual member of a swarm uses their own distinct personality and set of skills yet; by working together, they can address more difficult problems. This notion gave rise to a number of algorithms, each of which was named after a different natural phenomenon that served as its source of inspiration. As an illustration, Particle Swarm Optimization (PSO) algorithm, which was initially presented by Kennedy and Eberhart (1995), is designed to simulate the behaviour of groups of birds and fish who are seeking for food. Kevin M. Passino invented Bacterial Foraging Optimization (BFO) ("Biomimicry of Bacterial Foraging for Distributed Optimization and Control") (Passino, 2002), emulates the chemotactic behavior of bacteria like E. coli navigating their environment to optimize food search. Karaboga et al. (2014) developed the Artificial Bee Colony (ABC) algorithm in 2005, which simulates how honey bees use their food resources in a collaborative foraging approach. The Cuckoo Search (CS) method was developed by Yang and Deb (2009) and is based on the cuckoos' tendency to parasitise their brood. The Bat Algorithm (BA), developed by Yang (2010) is based on the echolocation behavior of bats when hunting for prey and commuting through a dark environment. The Whale Optimization Algorithm (WOA), is a newly developed optimization method inspired by the bubble-net hunting strategies of Humpback whales proposed by Mirjalili and Lewis (2016). Mirjalili et al. (2014) also developed the Grey Wolf Optimizer (GWO), inspired by the social hierarchy and leadership-based hunting strategy of grey wolves. Based on the opportunistic hunting behavior of red foxes, which is characterized by flexibility and resourcefulness, Połap and Woźniak (2021) introduced the Red Fox Optimization (RFO). Most recently, focussing on their coordination and communication in hunting prey, Chopra and Ansari (2022) Golden Jackal Optimization (GJO) algorithm, replicates the cooperative hunting strategies of golden jackals. These algorithms show how research on animal behaviour could inspire creative answers for challenging optimization challenges.

ii) *Evolutionary techniques*: Techniques within the domain of evolution are techniques that derive their inspiration from biological processes. These techniques are referred to as evolutionary techniques. Crossover and mutation are used in evolutionary algorithms to generate new solutions and eliminate bad ones to improve fitness. Over the period of their development, these methods have been significantly influenced by the contributions of a wide range of scholars. Evolutionary Programming ("Artificial Intelligence through Simulated Evolution," 2009), which replicates the evolution of behavioural strategies, was introduced by Fogel (1998). Evolutionary Programming is notably focused on finite-state machines. Genetics and natural selection served as inspiration for development of Genetic Algorithms (GA), which included mechanisms such as mutation, selection, and crossover. Differential Evolution (DE), was created by Storn and Price (1997). It focusses on differences between groups and mutations and crossovers. During the same year, Koza (1994) proposed Genetic Programming (GP), which focuses on the evolution of computer programs using genetic operators. More lately, Kiran (2015) introduced the Tree Seed Algorithm (TSA), a model that replicates the seed distribution and growth in trees. It emphasises diversity and exploration in evolution.

iii) *Physics-based techniques*: The fundamental principles that govern natural events serve as the foundation for physics-based methods. The metallurgical annealing process was the inspiration for the 1996

introduction of Simulated Annealing (SA) (Kirkpatrick et al., 1983). Rashedi et al. (2009) developed the Gravitational Search Algorithm (GSA) based on Newton's law of gravity. Hatamlou (2013) used the idea of black holes from astronomy to develop the Black Hole Algorithm (BHA). Hashim et al. (2021) developed the Archimedes Optimization Algorithm (AOA), based on Archimedes' principle.

iv) ***Human-related techniques draw inspiration from human behaviour*****:** Individuals engage in both physical activities affecting performance and non-physical activities like cognitive processes and behaviour. Teaching-Learning-Based Optimization (TLBO) was proposed by Rao et al. (2011) and is based on classroom interactions. Shi (2011) proposed Brain Storm Optimization (BSO), modeled after human brainstorming processes, focusing on idea generation and evolution. Inspired by the features that humans utilize both logical and illogical behaviors for decision-making, Ahmadi (2017) introduced Human Behavior-Based Optimization (HBBO) inspired from cognitive processes, particularly those used by human beings to solve problems and retrieve memory-based information and decision-making strategies, Mousavirad and Ebrahimpour-Komleh (2017) proposed Human Mental Search (HMS) framework in 2017. Teamwork Optimization (TO) was introduced by Dehghani and Trojovský (2021) to model the collaboration strategies used by humans during teamwork. In 2020, three researchers Askari et al. (2020) introduced the Political Optimizer (PO), they developed it based on political strategies and political behavior like forming alliances or competing.

The No-Free-Lunch (NFL) theorem asserts that no singular method is capable of resolving all optimization issues. This theory prompts several scholars to dedicate their efforts to developing new metaheuristics algorithm and improving existing techniques through parametric tuning. Researcher are putting a lot of effort into improving the existing algorithms rather than developing entirely new ones. It is possible to achieve improved performance and efficiency through the implementation of enhancements such as adaptive mechanisms, the fine-tuning of parameters, and the incorporation of intelligent techniques. Better accuracy, speed, and scalability may be achieved by using existing algorithms in this way and fixing their individual weaknesses.

In this paper, an adaptive population learning based improved variant of EJaya named “PLEJaya” has been proposed. EJaya is an enhanced version of Jaya algorithm proposed by Zhang et al. (2021) which improves global exploration of the Jaya algorithm. However, due to its single-learning approach and inadequate population maintenance, it also has several disadvantages, including slow convergence rates and stagnation at local minima. To enhance the exploration and exploitation capability of EJaya algorithm the Adaptive Enhancement technique is used. The given strategy is to let large changes in the initial stages to cover a large search area and progressively reduce the step size in following iterations to fine-tune the results. The Linear Population Reduction (LPR) approach is used to enhance convergence speed. This step seeks to decrease the population size, helping the algorithm to focus on the remaining optimal solutions as it converges. This population reduction approach uses greater computational power to locate better optimum solutions, and the search improves after population reduction. This combination allows us to enhance performance on difficult optimization problems, addressing challenges that EJaya algorithms may struggle with.

The performance of the proposed method will be assessed on various benchmark functions and real-world optimization issues, and compared to existing algorithms such as TLBO, EJAYA, Jaya, PSO, GA, AOA, GWO, and WOA. It will be shown that this approach works well at solving difficult optimization problems, which means it could be used in many other areas as well.

The portions of the paper are written as follows: A review of EJAYA is presented in Section 2, followed by an explanation of the recommended methodologies in Section 3, numerical experiments and comparisons in Section 4, and the conclusion of the paper in Section 5.

2. EJAYA Algorithm

The JAYA algorithm (Rao, 2016) is enhanced by the incorporation of a more sophisticated exploitation and exploration mechanism. In addition to the current best and worst solutions, this improvement is accomplished by taking into account the population's average. The algorithm's goal is to establish a balanced search by incorporating these factors, which helps to prevent premature convergence and improves the ability to escape local optima.

2.1 Key Features and Steps

i) *Initialization*: Population Size (N), upper bound (u), lower bound (l), Number of Variables (D), Current Evaluations (gen), and Maximum Evaluations (max Gen) are initialized. The initial population (X) and historical population (X^{old}) are randomly generated within the variable limits using Equation (1).

$$x_{ij} = l_j + (u_j - l_j) * rand, \text{ where } i = 1, 2, ..., N \ \& \ j = 1, 2, ..., D \quad (1)$$

ii) *Population evaluation*: The fitness of each individual in the population is calculated, and the current best solution X_{Best} is identified.

iii) *Update current evaluations*: gen is updated $gen = gen + N$.

iv) *Stopping condition*: If gen exceeds max Gen, the algorithm terminates, returning X_{Best}. Otherwise, the algorithm proceeds.

v) *Local exploitation and global exploration*: The chosen probability P_{select} is generated. If P_{select} is greater than 0.5, the method uses local exploitation strategy otherwise, global exploration strategy is performed.

a) Local exploitation:

Calculate Mean Solution (M):

$$M = \frac{1}{N} \sum_{j=1}^{N} x_j \quad (2)$$

where, N is the population size.

Calculate Upper and Lower Points:

$$P_u = \lambda_1 \times x_{Best} + (1 - \lambda_2) \times M \quad (3)$$

$$P_l = \lambda_3 \times x_{Worst} + (1 - \lambda_4) \times M \quad (4)$$

Update Position:

$$v_i = x_i + \lambda_5 \times (P_u - x_i) - \lambda_6 \times (P_l - x_i), \quad i = 1, 2, 3, ..., N \quad (5)$$

b) Global exploration:
A different mechanism can be used for exploration, typically involving historical populations or random distributions to explore new regions. The fundamental idea behind this global exploration approach is that the differential vectors between the historical population and the current population cover a bigger search area than those inside the same generation. To apply this concept, First, a random selection approach generates the historical population. After that, the historical population is randomly shuffled to reorganize the population.

The following is a description of this strategy:

Generate the historical population: The historical population, is created using the following rule:

$$X^{old} = \{x_1^{old}, x_2^{old}, ..., x_n^{old}\}$$

$$X^{old} = \begin{cases} X, & if\ P_{select} \le 0.5 \\ X^{old}, & otherwise \end{cases} \quad (6)$$

where, X is the current population, and P_{select} is the selected probability (a random number uniformly distributed between 0 and 1).

Shuffle the historical population: The historical population X^{old} is then shuffled using the following operation:

$$X^{old} = permuting(X^{old}) \quad (7)$$

where, permuting is a random shuffling function that rearranges all the individuals in X^{old} in random order.

Global exploration strategy: The global exploration strategy in EJAYA can be formulated as:

$$v_i = x_i + rand \times (x_i^{old} - x_i), \quad i = 1, 2, 3, ..., N \quad (8)$$

where, $rand$ is a random number.

vi) *Selection*: Accept v_i if it gives a better function value.

vii) Go to step 3.

3. Proposed Algorithm

To enhance the performance of EJaya, we suggest a three-phase optimization technique. The initial phase of algorithm is Adaptive Population Enhancement Phase to enhance the exploration and exploitation capability. After the completion of first phase the algorithm moves onto the Ejaya phase focused on refining and improving the solutions established in the preceding phase. This phase focuses on exploiting the most promising portions of the solution space while iteratively improving the quality of candidate solutions. In third phase, after each generation of the above two phases, the population size is decreased by using the linear population reduction mechanism which enhance the convergence speed.

The three phases of the method are described here:

Phase 1: Adaptive Population Enhancement Phase
During the Adaptive Population Enhancement Phase, the optimization method refines candidate solutions based on their relative fitness using an adjustable scaling factor S*F* which decrease as number of generations increases. The strategy is to let large changes in the initial stages to cover a large search area and progressively reduce the step size in following iterations to fine-tune the results.
Population update: In this phase, two individuals from the population, $x(a)$ and $x(b)$, are randomly chosen for comparison. The fitness values of these individuals, $f(a)$ and $f(b)$, dictate the update mechanism:

If f(a) < f(b) the solution is modified in a manner that improves its quality, prioritizing the superior individual $x(a)$:

$$v_i = X_{old,i} + SF \times (X_{a,i} - X_{b,i}) \tag{9}$$

This upgrade improves convergence towards better solutions by using knowledge from the more proficient solution.

If $f(a) > f(b)$ the update utilizes a different approach, amplifying the impact of $x(b)$ while considering its relative location to $x(a)$:

$$v_i = X_{old,i} + SF \times (X_{b,i} - X_{a,i}) \tag{10}$$

where,
The adaptive scaling factor F is delineated as follows:

$$\mathrm{SF} = \mathrm{SF}_1 - \left(SF_2 \times \frac{\mathrm{gen}}{\max Gen} \right) \tag{11}$$

This enables the step size to decrease during the iterations from 0.9 to 0.2. The adaptive factor SF was selected within the range 0.9 to 0.2 to gradually shift the algorithm from a predominantly exploratory phase to a more exploitative phase. Higher values of SF at the beginning increase the search radius and help in identifying diverse regions, while lower values toward the end allow finer exploitation around promising solutions. This behavior aligns with established adaptive strategies used in population-based algorithms.

This method ensures that, even if the first chosen superior solution is suboptimal, the algorithm persists in exploring the dynamics of the search space.

The Adaptive Enhancement Phase significantly improves the optimization process via a dynamic updating technique that facilitates convergence to optimum solutions while permitting varied exploratory motions throughout the solution space.

Phase 2: Ejaya Phase
Following the completion of Phase 1, the algorithm will then go on to Phase 2, where it will make use of EJAYA's techniques in order to further analyse and improve the solution space. A specific probability is generated to determine the choice between these strategies. The algorithm starts local exploitation when the probability surpasses 0.5. Equations (2), (3), (4), and (5) are used to refine solutions in this phase. Global exploration starts with probability 0.5 or below. Equations (6), (7), and (8) expand the solution space search in this method. This dual strategy improves the algorithm's capacity to identify optimal solutions by preserving population diversity while narrowing promising areas.

Phase 3: Linear Population Reduction

After each generation of the above phases, the Linear Population Reduction (LPR) mechanism is applied. As the algorithm converges, this phase aims to reduce the population size so that it can concentrate on the remaining optimum solutions and increase convergence speed ensuring that the algorithm can find the best solution faster. This method for reducing the population helps the program use more computing power to find better optimal solutions, while the search gets better as the population size goes down. The decrease keeps going until a certain stopping condition is met, which lets the algorithm truly get to the best answers.

Mathematical Formulation:

For each generation gen, the population size pop(gen) decreases linearly:

$$pop(gen) = pop_{initial} - \left(\frac{pop_{initial} - pop_{final}}{maxGen} \right) \times gen \tag{12}$$

where, $pop_{initial}$ is the initial population size, pop_{final} is the final population size, gen is current generation, *maxGen* is the maximum number of generations.

By making the population smaller over time, the LPR process helps the algorithm focus on fewer, better solutions, which makes it converge faster in the later stages. These three steps make sure that the algorithm does a good job of balancing exploration and exploitation, speeds up convergence, and keeps the population diverse during the optimization process.

These three steps make sure that the algorithm does a good job of balancing exploration and exploitation, speeds up convergence, and keeps the population diverse during the optimization process.

The pseudo-code of PLEJaya is given below:

Pseudocode of the Proposed PLEJaya Algorithm

Initialize Population (*X*) with size *N*, Variable Limits (*u,l*), Number of Variables (*D*), $pop_{initial}$, pop_{final} , Set Maximum Evaluations (*maxGen*), and Current Evaluations (*gen* = 0).

Compute the Initial Fitness of the Population using Equation (1).

Identify Best (x_{Best}) and Worst (x_{Worst}) Solutions

while $(gen < \max Gen)$

Phase 1: Adaptive Enhancement Phase

For $i = 1:N$

Randomly select two distinct solutions, $X_j \& X_k$, where $j \neq k$.

If $f(X_j) < f(X_k)$

$v_i = X_{old,i} + SF(X_j - X_k)$

Else
$v_i = X_{old,i} + SF(X_k - X_j)$

End if

End for

If $f(v_i) < f(X_i)$

$X_i = v_i$

Update Fitness of Population

Update Best $Best()$ and $Worst()$ solutions.

Phase 2: EJAYA Phase

The selected probability P_{select} is generated.

If

$P_{select} > 0.5$

The local exploitation strategy is performed using Equations (2), (3), (4), & (5)

Else

Global exploration strategy is performed using Equations (6), (7), & (8).

End if

$if\ f(v_i) < f(X_i)$

$X_i = v_i$

Update Fitness of Population

Phase 3: Linear Population Reduction

Sort the population

After sorting Apply Linear Population Reduction (LPR) mechanism using Equation (13)

Update $Best()$ and $Worst()$ solutions

$gen = gen + 1$

Return Best solution (x_{Best}).

4. Numerical Experiments and Comparisons

4.1 Experimental Settings

To evaluate the optimization efficiency and solution quality of the proposed PLEJaya algorithm, a comprehensive set of fifty-two benchmark functions is employed. Twenty-three standard objective functions—unimodal, high-dimensional multimodal, and fixed-dimensional multimodal—and thirty CEC 2017 test suite functions are included. The performance of the PLEJaya algorithm is compared against eight well-known metaheuristic algorithms: TLBO, EJAYA, JAYA, GA, PSO, AOA, GWO, and WOA. On each algorithm, thirty different runs of one thousand iterations are carried out.

The evaluation metrics used for comparison are five statistical indicators: mean, standard deviation (std), best, CPU time (CPT), and rank. Optimization algorithms are ranked by their mean rank across all objective functions. When the mean is same, the standard deviation is taken into account. The lowest standard deviation is prioritized first. When both the mean and standard deviation are identical, the method with the least CPU time is ranked first. **Table 1** displays the parameter settings for various methods.

Preliminary experiments were conducted with different SF bounds (e.g., 1.0-0.5, 0.8-0.1). The value 0.9 and 0.7 consistently produced stable convergence and balanced exploration-exploitation behaviour across most benchmark functions. Therefore, these bounds were selected for the final implementation.

Table 1. Parameter settings for various methods.

Algorithm	Parameters
PSO	Inertia weight(w) decreases linearly from 0.9 to 0.1 C1 (Personal learning coefficient) = 2 C2 (Global learning coefficient) = 2
GA	Mutation probability = 0.05 Crossover probability = 0.8 Selection = Roulette wheel (proportionate)
GWO	Convergence parameter (a) decreases linearly from 2 to 0
WOA	Convergence parameter (a) decreases linearly from 2 to 0 a2 linearly decreases from −1 to −2
AOA	control parameter (u) = 0.5 sensitive parameter (alpha) = 5
TLBO	Teaching factor (TF) = round [(1 + rand)]
PLEJaya	Population Size = 100 $SF_1 = 0.9$ $SF_2 = 0.7$ $pop_{initial} = 100$ $pop_{final} = 10$ $maxGen = 1000$

The benchmark functions are chosen based on the following reasons: Unimodal functions F1 to F7, are ideal for assessing the exploitation capabilities of metaheuristic algorithms, as they allow for efficient convergence to the global optimum without the interference of local optima. The exploration capabilities of these algorithms may be effectively measured by multimodal functions, such as F8 to F23, because of their many local optima. Metaheuristic algorithms' ability to strike a balance between the search's exploration and exploitation stages is tested using the CEC2017 test suite's sophisticated benchmark functions.

4.2 Unimodal Benchmark Problems

The testing of unimodal objective functions was done in order to evaluate the PLEJaya algorithm's capacity for exploitation. Unimodal functions were chosen because they contain only a single optimal solution and do not have local optima. The optimization results for the F1 to F7 functions using the PLEJaya and several competing algorithms are presented in **Table 2**. The outcomes indicate that the PLEJaya successfully reached the global optimum for all the objective functions. When compared to five other algorithms, PLEJaya demonstrated clear superiority and highly competitive performance.

Table 2. Evaluation results on unimodal functions.

		PLEJaya	TLBO	EJAYA	JAYA	PSO	GA
F1	Best	5.00E-25	1.5992	9.9608E-16	2.95632	3.14E−05	33.35136
	Mean	5.54E-24	2.055918	3.82E-15	4.487295	0.181599	19.78438
	STD	5.656E-24	0.520663	9.5587E-15	1.056918	0.729657	8.395292
	CPT	5.439462	13.286505	4.400951	2.595141	419.606604	6.644676
	Rank	1	4	2	5	3	6
F2	Best	1.20E-12	1.7577	8.22E-10	1.500848	0.080876	1.869152
	Mean	1.99E-12	2.055918	3.85E-09	1.980803	1.419726	3.188112
	STD	6.2125E-13	0.520663	2.5946E-09	0.436198	3.034488	0.693334
	CPT	7.420475	13.022129	5.027499	13.428985	171.197969	5.918492
	Rank	1	5	2	4	3	6
F3	Best	2.5281E-07	0.016224	0.005337	28652.11	29.63701	1111.139
	Mean	1.2575E-06	2279.459	0.041555	32331.69	1094.128	2125.752
	STD	8.9275E-07	1697.011	0.038809	4174.634	1618.416	495.7954
	CPT	13.340652	25.033064	14.110897	2.155619	126.053377	20.103473
	Rank	1	5	2	6	3	4
F4	Best	7.51E-06	2.9651	0.008997	17.153	3.511675	2.113563
	Mean	3.68E-05	3.236455	0.039634	21.77216	6.507623	3.210794
	STD	2.952E-05	0.302485	0.025256	2.994494	1.970793	0.64983
	CPT	6.468397	12.929743	4.602314	2.267819	109.659035	9.183299
	Rank	1	3	2	6	5	4
F5	Best	0	30.7532	10.8336	484.448	30.07967	227.4906
	Mean	0	33.813	29.04889	806.2018	112.2916	420.6
	STD	0	2.226827	23.07953	308.0335	85.244	122.0165
	CPT	9.147682	20.660965	11.235353	6.168703	119.717372	11.067501
	Rank	1	3	2	6	4	5
F6	Best	2.21E-24	0.13432	7.01E-17	9.7929	9.98E−06	14.51884
	Mean	1.40E-23	0.193379	2.27E-15	12.66313	0.028587	34.0323
	STD	1.012E-23	0.064751	2.635E-15	2.974	0.0747	15.199
	CPT	9.272193	12.733069	5.047241	8.121155	112.115750	10.359740
	Rank	1	4	2	5	3	6
F7	Best	0.001095	0.010721	0.007093	0.002763	0.082498	0.006283
	Mean	0.005133	0.021664	0.009197	0.005465	0.166943	0.009847
	STD	0.002111	0.005344	0.003695	0.001644	0.052796	0.003084
	CPT	9.296877	20.265117	8.842797	2.169309	114.826136	13.788341
	Rank	1	5	4	2	6	3
Sum Rank		7	29	16	34	27	34
Mean Rank		1	4.14	2.29	4.86	3.86	4.86
Total Rank		1	4	2	5	3	5

4.3 Multimodal Benchmark Problems

To determine the exploring capability of the "PLEJaya algorithm" the assessment of high-dimensional multimodal objective functions was carried out. High-dimensional multimodal objective functions were chosen because they have various local and global optima. The optimization results for the F8 to F13 functions using the PLEJaya and several competing algorithms are presented in **Table 3**. The "PLEJaya algorithm" is the top performer for optimizing all functions F8 to F13, according to the data, which indicates that it consistently converged to the global optimum for each of these functions.

Table 3. Evaluation results on multimodal functions.

		PLEJaya	TLBO	E-JAYA	JAYA	PSO	GA
F8	Best	-9919.69	-6601.16	-7352.15	-7340.19	− 8047.43	− 9693.59
	Mean	-8585.1	-5234.86	-6963.12	-8355.31	− 6891.6	− 8551.34
	STD	716.8452	499.5731	542.8759	688.8714	874.6514	761.0887
	CPT	4.595238	11.758071	2.348841	9.308895	117.963119	6.489410
	Rank	1	6	4	3	5	2
F9	Best	11.9395	192.9874	14.9244	18.9042	32.85466	32.80754
	Mean	16.1183	208.4517	22.88405	24.34615	69.57591	58.68141
	STD	3.3298	8.536852	7.706914	5.476159	20.94823	12.70118
	CPT	18.257091	19.390398	10.272244	7.760017	120.208005	9.984504
	Rank	1	6	2	3	5	4
F10	Best	2.42E-13	2.0201	9.95E-09	2.402	0.978948	3.045616
	Mean	2.74E-12	3.039636	2.09E-08	4.542082	2.869241	3.659085
	STD	2.31678E-12	0.876241	1.63E-08	5.133384	0.78912	0.411662
	CPT	14.797849	30.498326	11.305578	10.295386	120.919220	11.312803
	Rank	1	4	2	6	3	5
F11	Best	0	0.00675	1.89E-15	1.0245	0.012446	1.251305
	Mean	0	0.030281	0.005808	1.037572	0.308563	1.524898
	STD	0	0.032233	0.013367	0.016212	0.899622	0.143664
	CPT	12.171252	29.214839	8.060355	7.322093	517.012686	9.615891
	Rank	1	3	2	5	4	6
F12	Best	1.23E-26	0.18432	5.89E-17	4.5874	0.000558	0.042239
	Mean	8.62E-25	0.313035	1.03E-08	6.955473	1.391328	0.155349
	STD	1.8333E-24	0.101105	3.43008E-08	2.17984	1.03366	0.075924
	CPT	12.948925	31.106377	9.867746	10.992171	127.343985	10.394519
	Rank	1	4	2	6	5	3
F13	Best	5.07E-25	0.25983	2.39E-16	4.694	0.056865	0.993663
	Mean	1.25E-23	0.341814	2.00E-03	7.719255	3.08976	2.160287
	STD	1.2483E-23	0.075441	0.004444	2.618373	3.228647	0.701865
	CPT	12.267699	31.177067	9.111292	8.809444	123.644576	10.371147
	Rank	1	3	2	6	4	5
Sum Rank		8	26	13	28	26	25
Mean Rank		1.33	4.33	2.17	4.67	4.33	4.17
Total Rank		1	4	2	5	4	3

4.4 Fixed Multidimensional Benchmark Problems

Fixed-dimensional multimodal functions were used to evaluate PLEJaya's capacity to strike a balance between exploration and exploitation since they had fewer local optima than F8 to F13. **Table 4** presents the optimization outcomes for the F14 to F23 functions. Based on the results, the PLEJaya algorithm was shown to be the best optimizer for all functions from F14 to F23, constantly ranking top.

Table 4. Evaluation results on fixed multidimensional functions.

		PLEJaya	TLBO	E-JAYA	JAYA	PSO	GA
F14	Best	0.39789	0.998	0.730	0.998	0.998	0.998
	Mean	0.39789	0.998	0.730	0.998	3.212	0.998
	STD	0	6.96E−06	1.16441E-16	0	2.885	0.0002
	CPT	14.065655	63.548074	15.236534	24.212400	158.608970	23.734359
	Rank	1	4	2	3	6	5
F15	Best	0.00030	0.00030	0.00030	0.00034	0.00030	0.00077
	Mean	0.00030	0.00337	0.00039	0.000458	0.00164	0.01273
	STD	0	0.00733	9.35E-05	4.909E-05	0.00444	0.01058
	CPT	10.429765	33.285465	11.278019	11.372214	21.900404	11.474497
	Rank	1	4	2	3	5	6
F16	Best	-1.0316	− 1.0316	-1.0316	-1.0316	− 1.0316	− 1.0316
	Mean	-1.0316	− 1.0316	-1.0316	-1.0316	− 1.0316	− 1.0316
	STD	0	2.49E−06	4.57E-7	1.223E-05	1.14E−16	4.37E−06

Table 4 continued…

	CPT	9.420166	21.359316	8.694221	8.210418	21.880538	10.382517
	Rank	1	4	3	5	2	6
F17	Best	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789
	Mean	0.39789	0.39789	0.39789	0.39789	0.53901	0.52441
	STD	0	5.822E-17	5.822E-17	5.688E-17	0.539	0.534
	CPT	12.657265	37.556580	11.897511	10.668490	37.538695	12.439384
	Rank	1	2	2	3	5	4
F18	Best	3	3	3	3	3	3.000044
	Mean	3	3	3	3.001354	3	5.729191
	STD	0	0	0	1.694E-03	2.76E−15	8.39291
	CPT	12.657265	38.713201	12.188205	12.484914	13.816302	14.722119
	Rank	1	1	1	3	2	4
F19	Best	-3.8628	-3.8628	-1.8997	-3.8628	− 3.86278	− 3.86278
	Mean	-3.8628	-3.8628	-1.8997	-3.8628	− 3.86278	− 3.86278
	STD	0	0	4.658E-16	0	2.09E−15	0.001431
	CPT	11.188305	34.240394	11.369006	11.442573	19.398306	11.596128
	Rank	1	1	2	1	3	4
F20	Best	-3.322	-3.322	-3.322	-3.2031	− 3.322	− 3.3214
	Mean	-3.322	-3.322	-3.322	-3.26011	− 3.29822	− 3.19552
	STD	0	0	0	0.058921	0.048793	0.093531
	CPT	11.281090	33.154432	10.864058	11.349633	27.081922	11.364303
	Rank	1	1	1	3	2	4
F21	Best	-10.1532	-10.1532	-10.1532	-9.8179	− 10.1532	− 9.0381
	Mean	-10.1532	-10.1532	-10.1532	-6.66069	− 5.77879	− 5.89083
	STD	0	0	0	2.15082	3.703566	2.512564
	CPT	11.582073	34.604902	10.683051	10.638662	31.327449	12.635227
	Rank	1	1	1	2	4	3
F22	Best	-10.4029	-10.4029	-10.4020	-10.4029	− 10.4029	− 10.1952
	Mean	-10.4029	-10.4029	-10.4029	-8.98973	− 6.31807	− 7.21825
	STD	0	0	0	2.08157	3.837031	2.472441
	CPT	11.531746	33.852822	10.567918	10.070323	52.636237	11.807742
	Rank	1	1	1	2	4	3
F23	Best	-10.5364	-10.5364	-10.5364	-10.5364	− 10.5364	− 10.417
	Mean	-10.5364	-10.5364	-10.5364	-8.72248	− 5.62285	− 5.78525
	STD	0	0	0	2.016805	3.755817	2.966829
	CPT	11.997129	35.719880	11.310603	10.699930	27.165628	12.041502
	Rank	1	1	1	2	4	3
Sum Rank		10	20	16	27	37	42
Mean Rank		1	2	1.6	2.7	3.7	4.2
Total Rank		1	3	2	4	5	6

4.5 Evaluation of the CEC 2017 Test Suite Benchmark Functions

In this section, we test the ability of the proposed algorithm in resolving the complex optimization problems that are introduced in the CEC 2017 test functions. These functions are categorized as follows: three unimodal functions (C17-F1 to C17-F3), seven multimodal functions (C17-F4 to C17-F10), ten hybrid functions (C17-F11 to C17-F20), and ten composition functions (C17-F21 to C17-F30).

Results from optimizing the CEC 2017 functions using the PLEJaya and alternative competing algorithms are displayed in **Table 5**. Based on the findings, it is evident that the PLEJaya algorithm obtains the highest level of performance when it comes to solving the functions C17-F1 to C17-F19, C17-F21, C17-F22, C17-F24, C17-F25, and C17-F27 to C17-F30.

Results of these findings reveals that the suggested strategy improves most CEC 2017 benchmark functions. Thus, PLEJaya solves optimization issues better than competitors.

Table 5. Evaluation results on CEC 2017 test functions.

		PLEJaya	TLBO	EJAYA	JAYA	PSO	GA	AOA	GWO	WOA
C17-F1	Best	1.08E+02	1.97E+07	1.14E+02	9.09E+03	7.27E+06	1.52E+07	2.57E+05	1.49E+03	5.34E+05
	Mean	1.55E+02	2.24E+07	1.88E+02	7.59E+02	1.05E+07	2.26E+07	1.28E+05	1.51E+03	4.86E+08
	STD	3.25E+01	3.70E+06	1.04E+02	1.18E+04	1.38E+07	1.05E+07	9.14E+04	2.51E+01	7.56E+07
	CPT	20.80	45.61	31.41	19.05	284.53	41.99	23.76	26.56	69.27
	Rank	1	8	2	3	6	7	5	4	9
C17-F2	Best	2.00E+02	1.18E+30	2.66E+02	1.65E+03	1.56E+03	5.32E+20	3.98E+17	1.58E+20	1.65E+29
	Mean	2.18E+02	8.53E+29	7.87E+04	2.28E+03	1.88E+03	4.65E+22	4.67E+17	1.81E+20	2.26E+30
	STD	2.69E+01	1.62E+28	1.11E+05	8.93E+02	1.68E+02	6.50E+22	3.23E+18	4.21E+20	6.73E+29
	CPT	14.92	51.36	31.56	16.25	325.77	25.59	14.99	19.77	68.36
	Rank	1	7	4	3	2	6	5	8	9
C17-F3	Best	3.00E+02	1.07E+05	5.74E+02	2.93E+04	8.43E+04	3.13E+04	8.14E+04	1.98E+04	4.56E+04
	Mean	3.00E+02	1.35E+05	1.14E+03	3.82E+04	5.69E+04	3.25E+04	9.18E+04	1.98E+04	3.26E+05
	STD	0	2.61E+04	7.14E+02	1.26E+04	1.69E+03	1.79E+03	1.21E+04	2.49E+02	6.73E+04
	CPT	22.30	46.93	28.29	14.46	423.13	26.70	18.48	20.10	45.77
	Rank	1	8	2	4	5	3	6	7	9
C17-F4	Best	4.05E+02	4.94E+02	4.26E+02	4.72E+02	6.16E+02	5.32E+02	5.10E+02	5.29E+02	5.43E+02
	Mean	4.57E+02	4.96E+02	4.57E+02	4.96E+02	6.27E+02	5.43E+02	5.18E+02	7.30E+02	7.62E+02
	STD	1.49E+01	1.92E+01	4.32E+01	3.46E+01	6.15E+02	1.55E+01	1.56E+01	3.45E+02	1.74E+02
	CPT	15.56	46.46	30.01	19.02	432.03	26.18	15.28	21.04	62.46
	Rank	1	4	2	3	7	6	5	8	9
C17-F5	Best	5.43E+02	7.11E+02	5.34E+02	6.16E+02	7.13E+02	7.21E+02	8.35E+02	5.90E+02	7.43E+02
	Mean	5.46E+02	7.23E+02	5.51E+02	6.32E+02	7.24E+02	7.22E+02	8.65E+02	5.91E+02	8.39E+02
	STD	4.22E+00	1.68E+01	2.39E+01	2.27E+01	1.51E+01	1.01E+00	2.59E+01	2.11E+00	5.25E+01
	CPT	20.75	53.39	33.33	14.93	434.45	29.45	36.32	25.05	45.83
	Rank	1	6	2	4	7	5	9	3	8
C17-F6	Best	6.00E+02	6.13E+02	6.00E+02	6.13E+02	6.11E+02	1.25E+04	6.09E+02	6.03E+02	6.23E+02
	Mean	6.00E+02	6.12E+02	6.00E+02	6.12E+02	6.31E+02	1.28E+05	6.90E+02	6.14E+02	6.67E+02
	STD	8.86E-05	4.72E−01	5.34E-02	0.54E+01	1.17E+01	2.12E+05	1.94E+00	0.074E+00	1.51E+01
	CPT	17.77	163.35	135.09	18.59	331.33	40.85	19.85	26.01	45.46
	Rank	1	3	2	4	6	9	8	5	7
C17-F7	Best	7.80E+02	9.14E+02	7.80E+02	9.33E+02	9.02E+02	1.35E+03	1.04E+03	8.07E+02	1.45E+03
	Mean	7.88E+02	9.64E+02	7.91E+02	9.37E+02	9.60E+02	1.35E+03	1.41E+03	8.69E+02	1.36E+03
	STD	1.01E+01	1.81E+01	1.54E+01	5.88E+00	8.20E+01	4.75E+00	3.15E+01	4.89E+01	6.97E+01
	CPT	20.53	56.90	36.74	15.25	327.48	29.96	28.35	25.72	43.61
	Rank	1	6	2	4	5	7	9	3	8
C17-F8	Best	8.32E+02	1.00E+03	8.51E+02	8.79E+02	9.27E+02	8.96E+02	1.32E+03	8.89E+02	1.01E+03
	Mean	8.39E+02	1.02E+03	8.78E+02	9.01E+02	9.52E+02	9.27E+02	1.57E+03	9.05E+02	1.05E+03
	STD	9.85E+00	8.05E+00	3.66E+01	3.17E+01	3.49E+01	4.34E+01	1.86E+01	2.36E+01	5.29E+01
	CPT	19.52	53.42	35.17	19.38	304.67	28.54	12.29	14.75	37.86
	Rank	1	7	2	3	6	5	9	4	8
C17-F9	Best	9.00E+02	2.16E+03	9.27E+02	2.02E+03	9.42E+02	5.79E+03	1.25E+03	1.08E+03	9.98E+03
	Mean	9.00E+02	2.29E+03	9.68E+02	2.77E+03	9.35E+03	7.47E+03	1.49E+03	2.59E+03	1.13E+04
	STD	0	1.80E+02	5.81E+01	1.05E+03	3.61E+02	2.37E+03	1.89E+02	1.32E+03	4.57E+03
	CPT	18.26	55.24	35.91	14.28	325.91	27.05	13.04	19.72	38.76
	Rank	1	4	2	6	8	7	3	5	9
C17-F10	Best	4.19E+03	7.74E+03	4.36E+03	4.20E+03	5.03E+03	5.50E+03	8.42E+03	3.42E+03	6.83E+03
	Mean	4.37E+03	7.96E+03	5.83E+03	4.98E+03	5.41E+03	6.07E+03	8.78E+03	4.41E+03	7.12E+03
	STD	2.63E+02	3.08E+02	2.08E+03	1.13E+03	5.31E+02	8.01E+02	3.42E+02	8.69E+02	7.15E+02
	CPT	14.22	111.71	83.58	15.11	323.74	33.63	17.12	19.12	42.35
	Rank	1	7	5	3	4	6	9	2	8
C17-F11	Best	1.12E+03	1.41E+03	1.15E+03	1.15E+03	1.17E+03	1.22E+03	1.26E+03	1.37E+03	6.75E+03
	Mean	1.13E+03	1.43E+03	1.15E+03	1.17E+03	1.24E+03	1.26E+03	1.33E+03	2.08E+03	6.99E+03
	STD	2.82E+00	2.84E+01	2.78E+00	5.32E+01	8.87E+01	5.77E+01	4.26E+01	6.50E+02	4.20E+03
	CPT	26.34	50.42	32.39	18.25	328.80	26.62	17.45	19.93	38.45
	Rank	1	7	2	3	4	5	6	8	9
C17-F12	Best	3.92E+03	4.94E+07	1.80E+04	4.27E+04	3.82E+04	3.69E+07	1.16E+05	7.62E+06	9.98
	Mean	3.72E+03	5.03E+07	3.96E+04	4.70E+04	1.24E+06	3.75E+07	4.67E+05	9.45E+07	2.15E+08
	STD	2.75E+03	1.39E+06	3.05E+04	6.07E+03	1.92E+06	6.83E+05	3.67E+05	1.60E+08	3.50E+08
	CPT	25.13	75.54	55.99	15.71	322.58	26.34	11.69	19.06	47.76

Table 5 continued…

	Rank	1	7	2	3	5	6	4	8	9
C17-F13	Best	1.78E+03	6.67E+05	1.47E+04	1.82E+03	4.63E+03	1.00E+06	1.07E+04	2.57E+07	1.35E+06
	Mean	3.73E+03	6.78E+05	3.42E+04	4.66E+03	1.24E+04	1.16E+06	1.73E+04	4.32E+07	1.97E+06
	STD	2.75E+03	1.50E+04	2.39E+04	4.01E+03	6.58E+03	8.27E+04	8.03E+04	1.83E+08	1.77E+06
	CPT	28.87	54.04	36.52	19.08	362.91	40.08	20.21	19.27	42.54
	Rank	1	6	5	2	3	7	4	9	8
C17-F14	Best	1.50E+03	3.58E+04	1.56E+03	5.34E+03	2.01E+03	3.57E+04	9.62E+04	4.22E+04	1.23E+06
	Mean	1.54E+03	3.77E+04	1.56E+03	1.28E+04	2.55E+03	5.99E+04	1.28E+04	4.56E+05	2.03E+06
	STD	6.16E+01	2.69E+03	3.56E+00	1.12E+04	3.95E+02	3.42E+04	3.22E+03	6.19E+05	1.39E+06
	CPT	25.282	78.933	56.244	17.740	237.015	40.692	11.392	29.15	39.97
	Rank	1	6	2	5	3	7	4	8	9
C17-F15	Best	1.74E+03	1.04E+05	4.84E+03	6.55E+03	1.95E+03	2.88E+04	1.58E+03	3.45E+05	8.76E+05
	Mean	1.85E+03	1.12E+05	5.85E+03	2.44E+04	5.27E+03	5.86E+04	2.21E+03	8.90E+05	1.29E+06
	STD	1.55E+02	1.29E+04	1.42E+03	2.53E+04	2.67E+03	4.21E+04	6.14E+02	1.32E+06	3.31E+06
	CPT	25.42	54.12	34.50	15.47	320.78	34.86	27.18	29.82	38.34
	Rank	1	7	4	5	3	6	2	8	9
C17-F16	Best	1.97E+03	3.30E+03	2.13E+03	2.68E+03	3.47E+03	3.37E+03	2.82E+03	2.39E+03	3.96E+03
	Mean	1.97E+03	3.33E+03	2.16E+03	2.85E+03	3.38E+03	3.40E+03	3.01E+03	2.75E+03	4.00E+03
	STD	1.89E+00	5.18E+01	9.30E+01	4.89E+02	1.13E+02	4.56E+01	1.76E+02	1.83E+02	6.14E+02
	CPT	40.19	76.43	51.53	60.15	327.13	27.37	47.73	48.82	54.75
	Rank	1	6	2	4	7	8	5	3	9
C17-F17	Best	1.71E+03	2.18E+03	1.77E+03	1.94E+03	2.21E+03	2.32E+03	1.95E+03	1.98E+03	1.76E+03
	Mean	1.74E+03	2.33E+03	1.86E+03	2.05E+03	2.55E+03	2.34E+03	1.99E+03	2.70E+03	2.86E+03
	STD	5.30E+01	2.03E+02	1.16E+02	1.58E+02	4.84E+02	4.69E+01	3.48E+01	1.07E+02	3.22E+02
	CPT	25.608	87.707	63.764	16.240	324.704	29.031	16.706	28.372	69.45
	Rank	1	5	2	4	7	6	3	8	9
C17-F18	Best	8.64E+03	1.11E+06	1.44E+04	3.25E+05	1.73E+05	8.72E+05	5.14E+05	1.02E+06	1.03E+07
	Mean	8.88E+03	1.15E+06	1.99E+04	4.11E+05	1.25E+07	8.90E+05	1.13E+06	1.42E+06	1.23E+07
	STD	3.43E+02	4.58E+04	7.90E+03	1.22E+05	1.75E+07	2.63E+04	7.89E+05	1.18E+06	1.28E+07
	CPT	23.77	60.34	40.17	14.42	322.88	32.11	27.14	27.84	38.18
	Rank	1	6	2	3	9	4	5	7	8
C17-F19	Best	2.06E+03	6.49E+04	2.97E+03	2.73E+03	3.26E+04	3.25E+06	8.99E+03	6.46E+04	9.42E+06
	Mean	2.18E+03	1.02E+05	7.01E+03	1.11E+04	6.17E+04	4.05E+06	1.07E+04	9.85E+05	1.82E+07
	STD	1.76E+02	5.25E+04	7.13E+03	1.29E+04	1.07E+05	1.12E+05	6.74E+03	1.59E+06	1.75E+07
	CPT	18.24	253.83	203.07	19.28	317.88	37.78	24.01	19.56	80.45
	Rank	1	6	2	4	5	8	3	7	9
C17-F20	Best	2.14E+03	2.62E+03	2.13E+03	2.24E+03	2.54E+03	2.66E+03	2.45E+03	2.26E+03	2.35E+03
	Mean	2.24E+03	2.74E+03	2.18E+03	2.36E+03	2.60E+03	2.69E+03	2.58E+03	2.59E+03	2.79E+03
	STD	1.44E+02	1.77E+02	6.35E+01	1.66E+02	8.89E+01	3.86E+01	1.05E+02	1.82E+02	2.24E+02
	CPT	17.10	105.43	82.21	15.15	336.04	30.43	19.244	19.995	38.85
	Rank	2	8	1	3	6	7	4	5	9
C17-F21	Best	2.32E+03	2.54E+03	2.37E+03	2.42E+03	4.57E+03	2.56E+03	2.43E+03	2.36E+03	2.40E+03
	Mean	2.32E+03	2.54E+03	2.37E+03	2.45E+03	8.72E+03	2.58E+03	2.46E+03	2.51E+03	2.65E+03
	STD	1.23E+01	2.25E+00	4.95E+00	7.69E+01	2.76E+03	1.25E+01	8.63E+01	3.21E+01	5.09E+01
	CPT	18.361	232.820	184.971	18.371	325.474	34.064	15.4856	23.242	120.35
	Rank	1	6	2	3	9	7	4	5	8
C17-F22	Best	2.30E+03	2.38E+03	2.30E+03	2.30E+03	2.31E+03	6.39E+03	2.31E+03	5.31E+03	7.41E+03
	Mean	2.30E+03	2.38E+03	2.30E+03	2.43E+03	2.96E+03	6.64E+03	2.32E+03	5.61E+03	7.89E+03
	STD	2.79E-09	4.48E+00	2.26E-08	1.85E+02	6.56E+01	3.55E+02	1.09E+01	2.05E+03	1.51E+03
	CPT	18.887	352.216	216.318	22.501	327.538	23.774	15.709	23.65	54.76
	Rank	1	4	2	5	6	8	3	7	9
C17-F23	Best	2.69E+03	2.79E+03	2.71E+03	2.30E+03	3.54E+03	3.35E+03	2.79E+03	2.81E+03	3.01E+03
	Mean	2.70E+03	2.86E+03	2.72E+03	2.30E+03	3.67E+03	3.36E+03	2.79E+03	2.89E+03	3.12E+03
	STD	1.61E+01	4.66E+00	9.25E+00	8.55E-07	3.90E+01	2.32E+01	8.98E+00	4.94E+01	6.67E+01
	CPT	27.876	322.157	260.549	19.266	398.648	30.487	16.465	34.72	76.54
	Rank	2	4	3	1	9	8	5	6	7
C17-F24	Best	2.86E+03	3.03E+03	2.89E+03	2.88E+03	3.13E+03	3.53E+03	2.93E+03	2.95E+03	3.05E+03
	Mean	2.86E+03	3.03E+03	2.90E+03	2.96E+03	3.19E+03	3.55E+03	2.97E+03	3.20E+03	3.29E+03
	STD	5.42E-01	3.86E+00	2.78E+01	4.595E+01	8.49E+01	3.32E+01	1.37E+01	7.18E+01	8.21E+01
	CPT	17.467	314.314	243.009	20.893	330.424	29.934	17.474	21.22	45.76

Table 5 continued…

	Rank	1	5	2	3	6	9	4	7	8
C17-F25	Best	2.73E+03	2.90E+03	2.89E+03	2.89E+03	3.00E+03	2.94E+03	2.89E+03	3.03E+03	3.10E+03
	Mean	2.73E+03	2.93E+03	2.89E+03	2.89E+03	3.02E+03	2.95E+03	2.98E+03	3.05E+03	3.12E+03
	STD	2.96E-02	2.38E+01	0.77E+00	0.01E+00	3.55E+01	1.09E+01	8.81E+01	6.13E+01	7.69E+01
	CPT	22.68	309.89	247.26	21.69	131.78	29.92	20.69	33.01	53.91
	Rank	1	4	3	2	7	5	6	8	9
C17-F26	Best	3.79E+03	4.01E+03	2.90E+03	5.07E+03	5.28E+03	8.82E+03	4.64E+03	4.86E+03	7.89E+03
	Mean	4.06E+03	4.98E+03	3.68E+03	5.27E+03	6.06E+03	9.02E+03	5.02E+03	4.99E+03	8.25E+03
	STD	3.87E+02	1.39E+03	1.10E+03	2.82E+02	9.42E+02	4.03E+02	2.61E+02	4.35E+02	9.85E+02
	CPT	17.41	403.67	313.15	18.01	443.84	30.05	17.93	33.45	63.84
	Rank	2	3	1	6	7	8	5	4	9
C17-F27	Best	3.19E+03	3.23E+03	3.21E+03	3.22E+03	3.19E+03	4.25E+03	3.23E+03	3.25E+03	3.26E+03
	Mean	3.20E+03	3.23E+03	3.22E+03	3.23E+03	3.42E+03	4.34E+03	3.24E+03	3.32E+03	3.54E+03
	STD	1.70E+01	5.03E-01	6.51E+00	1.59E+01	7.71E+01	1.37E+02	6.58E+00	2.14E+01	1.03E+02
	CPT	25.849	469.89	379.91	20.92	432.89	31.11	20.36	32.76	73.54
	Rank	1	3	2	4	7	9	5	6	8
C17-28	Best	3.10E+03	3.25E+03	3.22E+03	3.10E+03	3.12E+03	3.30E+03	3.23E+03	3.28E+03	3.48E+03
	Mean	3.18E+03	3.26E+03	3.24E+03	3.18E+03	3.37E+03	3.30E+03	3.49E+03	3.54E+03	3.67E+03
	STD	9.56E+01	1.31E+02	3.37E+01	1.06E+02	1.59E+02	1.49E+01	2.26E+01	1.75E+02	1.47E+02
	CPT	18.49	389.51	309.39	21.72	431.56	27.38	20.58	28.94	69.76
	Rank	1	4	3	2	6	5	7	8	9
C17-29	Best	3.36E+03	4.18E+03	3.37E+03	3.82E+03	3.30E+03	5.14E+03	3.91E+03	3.45E+03	4.59E+03
	Mean	3.40E+03	4.19E+03	3.67E+03	3.84E+03	4.48E+03	5.35E+03	3.96E+03	3.95E+03	5.18E+03
	STD	5.36E+01	9.86E+00	41.5E+02	2.94E+01	3.35E+02	2.92E+02	8.86E+01	2.25E+02	5.69E+02
	CPT	25.10	343.59	213.73	19.64	459.62	33.43	28.02	31.35	62.67
	Rank	1	6	2	3	7	9	5	4	8
C17-30	Best	6.18E+03	3.85E+05	6.63E+03	6.53E+03	9.89E+04	1.40E+05	8.59E+03	7.46E+06	9.78E+06
	Mean	6.37E+03	5.14E+05	9.44E+03	6.72E+03	5.32E+05	1.13E+06	1.56E+04	1.26E+07	6.51E+07
	STD	2.32E+02	1.83E+05	3.97E+03	2.68E+02	5.90E+05	1.11E+06	5.23E+03	1.45E+07	4.96E+07
	CPT	17.07	533.03	364.78	20.49	431.99	27.42	18.85	20.76	74.98
	Rank	1	5	3	2	6	7	4	8	9
	Sum Rank	33	168	70	104	178	200	152	183	256
	Mean Rank	1.1	5.6	2.3	3.47	5.93	6.67	5.07	6.1	8.53
	Total Rank	1	4	2	3	6	8	5	7	9

4.6 Evaluation Results on Real-World Problems

This section tests the PLEJaya algorithm's ability to resolve real-life optimization problems. Testing was done on four different engineering optimization problems using PLEJaya and other competing algorithms. These challenges include Welded Beam Design (WBD), Speed Reducer Design (SRD), Pressure Vessel Design (PVD), and Tension/Compression Spring Design (TCSD).

In **Table 6** the optimization results of four engineering problems are summarized. The simulation results demonstrate that PLEJaya consistently outperforms competitor algorithms in optimizing all four studied engineering challenges. The results clearly show that PLEJaya exhibits superior performance and robustness when dealing with real-world optimization problems.

PLEJaya provided the best results for the Welded Beam Design (WBD) issue, tied with E-JAYA and JAYA. Additionally, it maintained a reduced standard deviation (STD), which indicates that it is reliable. Similarly, in the PVD (Pressure Vessel Design) and Speed Reducer Design (SRD) problems, PLEJaya performed equally well with E-JAYA and JAYA but ranked higher due to its more consistent results. proposed

algorithm also placed first with the lowest standard deviation in the TCSD problem, proving its effectiveness across all challenges.

The proposed algorithm is the top-ranked algorithm in the context of these real-world engineering problems, surpassing other algorithms such as TLBO, E-JAYA, JAYA, PSO, and GA, as determined by the sum and mean of ranks.

Table 6. Evaluation results on real-world problems.

		PLEJaya	TLBO	E-JAYA	JAYA	PSO	GA
WBD	Best	1.724852	2.17541	1.724852	1.724852	1.876176	1.83841
	Mean	1.724852	2.54876	1.724852	1.724852	2.123005	1.96595
	STD	6.98647E-16	0.25631	9.98647E-15	4.65765E-14	0.034882	0.139733
	Rank	1	5	2	3	4	3
SRD	Best	2994.471	3033.594	2994.471	2994.471	3054.173	3070.629
	Mean	2994.471	3069.904	2994.471	2994.471	3174.457	3190.666
	STD	4.76943E-13	18.0977	4.76943E-13	4.76943E-13	92.69298	17.14086
	Rank	1	2	1	1	3	4
PVD	Best	5885.333	6166.438	5885.333	5885.333	5918.224	6581.043
	Mean	5885.333	6328.261	5885.333	5885.333	6265.49	6645.562
	STD	9.53886E-13	126.639	9.53886E-13	9.53886E-13	496.2457	657.679
	Rank	1	3	1	1	2	4
TCSD	Best	0.012665	0.012822	0.012665	0.012721	0.013151	0.013192
	Mean	0.012665	0.01296	0.012665	0.012721	0.014166	0.01289
	STD	0	0.007831	5.4272E-11	1.81939E-18	0.002092	0.000378
	Rank	1	5	2	3	6	4
Sum Rank		4	15	6	8	15	15
Mean Rank		1	3.75	1.5	2	3.75	3.75
Total Rank		1	4	2	3	4	4

4.7 Statistical Analysis

The performance of the PLEJaya was compared against five well-known algorithms TLBO, EJAYA, JAYA, PSO, and GA using statistical analysis methods. The Wilcoxon Rank-Sum test and the Friedman test were employed to assess the significance of PLEJaya's performance over classical functions and CEC17 functions are presented in **Tables 7**, **8**, **9**, and **10**.

Table 7. Wilcoxon-rank sum test results for classical functions.

Algorithms		Win/loss/ties	ΣR^+	ΣR^-	*z*-value	*p*-value	Sig at $\alpha = 0.05$
PLEJaya *vs*	TLBO	15/0/8	120	0	3.408	<0.001	+
	EJAYA	15/1/7	122	14	2.792	<0.001	+
	JAYA	19/1/3	197	13	3.435	<0.001	+
	PSO	21/0/2	231	0	4.015	<0.001	+
	GA	22/0/1	253	0	4.107	<0.001	+

The symbols "+" and "=" stand for better and equal respectively.

Table 8. Rankings according to Friedman for classical functions.

	PLEJaya	TLBO	EJAYA	JAYA	PSO	GA
Rank	1.54	3.63	2.37	4.30	4.46	4.70

Table 9. Wilcoxon-rank sum test results for CEC17 functions.

Algorithms		ΣR^+	ΣR^-	*z* value	*p*-value	Sig at α=0.01
PLEjaya *vs*	TLBO	465	0	4.782	<0.001	+
	Ejaya	348.5	29.5	3.834	<0.001	+
	Jaya	420	15	4.379	<0.001	+
	PSO	465	0	4.782	<0.001	+
	GA	465	0	4.782	<0.001	+
	AOA	465	0	4.782	<0.001	+
	GWO	465	0	4.782	<0.001	+
	WOA	465	0	4.782	<0.001	+

The symbols "+" and "=" stand for better and equal respectively.

Table 10. Friedman ranks and Bonferroni-Dunn's CD values for CEC17 functions.

Algorithm	Rank
PLEjaya	**1.17**
TLBO	5.68
Ejaya	2.33
Jaya	3.47
PSO	5.97
GA	6.80
AOA	5.22
GWO	5.90
WOA	8.47
CD (Level=10%)	1.7664
CD (Level=5%)	1.9262

4.7.1 Wilcoxon-Rank Sum Test

Using a non-parametric statistical technique called the Wilcoxon Rank-Sum test, the given method was compared to different optimization algorithms on various test functions. This test evaluates the ranks of two data sets to determine if there is a significant difference between them. The outcomes are summed up as follows: wins, losses, and ties denote the frequency at which the PLEJaya outperformed, underperformed, or performed equally compared to each method. For instance, the proposed algorithm proved to be superior than TLBO when it obtained 15 wins, 0 loses, and 8 ties. Furthermore, the PLEJaya's wins versus TLBO had a sum of ranks (ΣR+) of 120, while its losses (ΣR-) had a sum of ranks of 0, indicating no loss. The z-value, a standardized test statistic, which measures how far the observed rank difference is from the null hypothesis (no difference between the algorithms). A higher absolute value indicates a more significant difference. Which is 3.408 against TLBO, which reflects a significant difference in performance. Furthermore, the p-value, which indicates the likelihood that the results occurred by chance, was less than 0.001 for all comparisons. This consistently low p-value suggests strong evidence that the PLEJaya significantly outperforms the others. The '+' symbol in the table confirms that the PLEJaya is significantly better than all comparison algorithms at a significance level of α=0.05.

4.7.2 Friedman Rank Test

For the purpose of determining how well the algorithms performed over a wide variety of functions, the Friedman test was utilised in order to carry out an evaluation of the algorithms. This non-parametric test is performed with the intention of determining whether or not there are differences between groups that are comparable to one another. Based on the findings, it was determined that the PLEJaya had the lowest average rank (1.54), which indicates that it performed the best overall. TLBO received a value of 3.63, while EJAYA attained a level of 2.37. On the other hand, GA had the poorest performance, attaining a

rating of 4.70. The results indicate that the proposed algorithm consistently outperformed the other methods throughout the test functions.

4.8 Convergence Graph

Here we show the convergence graphs that were used to compare PLEJaya algorithm's performance to other algorithms on different benchmark functions. By graphically representing the optimization issue and showing how each method approaches the ideal solution over iterations, the convergence graphs demonstrate the algorithms' usefulness.

The graphs of the four standard test functions- F1, F6, F11, and F16 and for functions from CEC17: CEC17-F1, CEC17-F10, CEC17-F18, and CEC17-F30 —show convergence patterns in **Figures 1, 2, 3, 4, 5, 6, 7,** and **8**. The convergence patterns for each function illustrate PLEJaya's performance compared to TLBO, E-JAYA, JAYA, PSO, and GA. Also, the Bar chart presentation of "Friedman Ranks and Bonferroni-Dunn test" for CEC17 functions shows in **Figure 9**.

Results show that the proposed algorithm converges quicker and more consistently to the optimum solution. The convergence graphs show that the proposed technique outperforms.

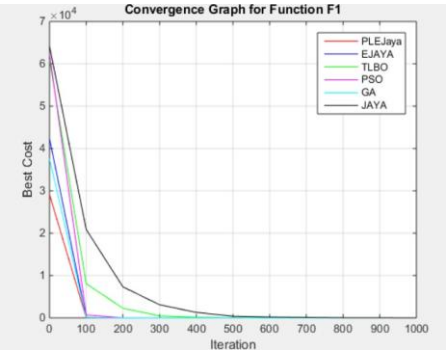

Figure 1. Convergence graph for F1.

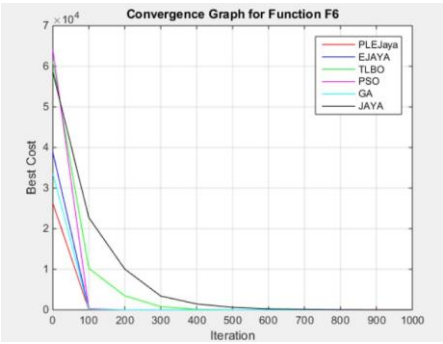

Figure 2. Convergence graph for F6.

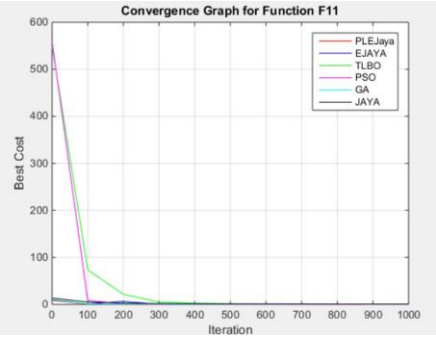

Figure 3. Convergence graph for F11.

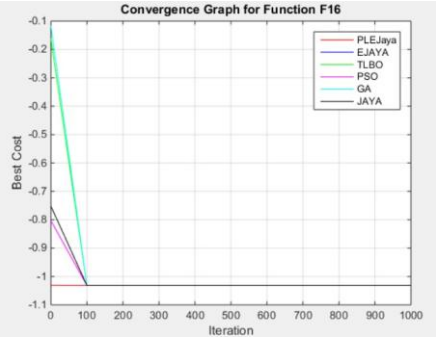

Figure 4. Convergence graph for F16.

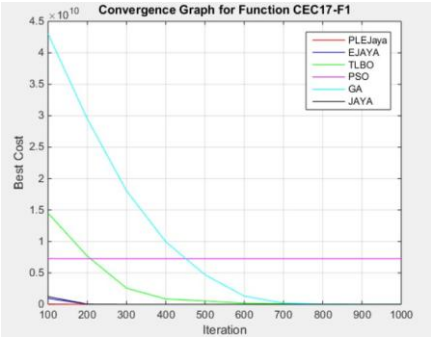

Figure 5. Convergence graph for CEC17-F1.

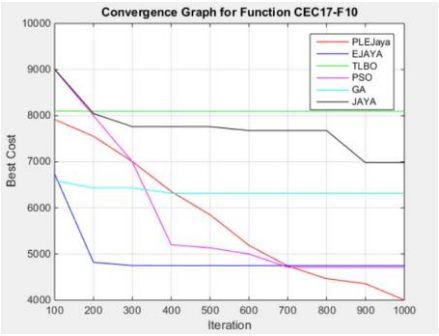

Figure 6. Convergence graph for CEC17-F10.

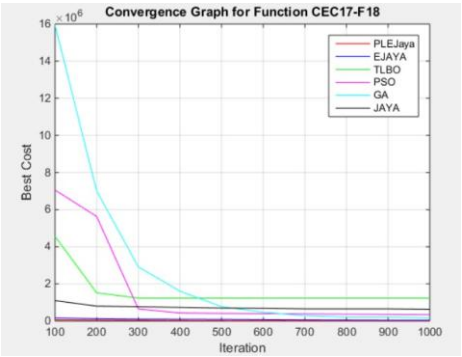

Figure 7. Convergence graph for CEC17-F18.

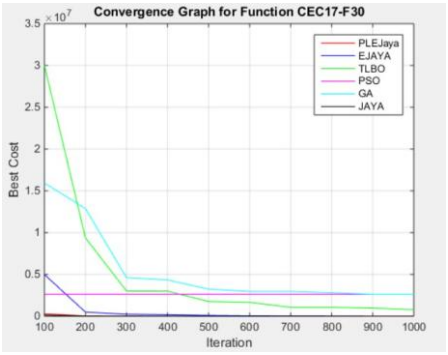

Figure 8. Convergence graph for CEC17-F30.

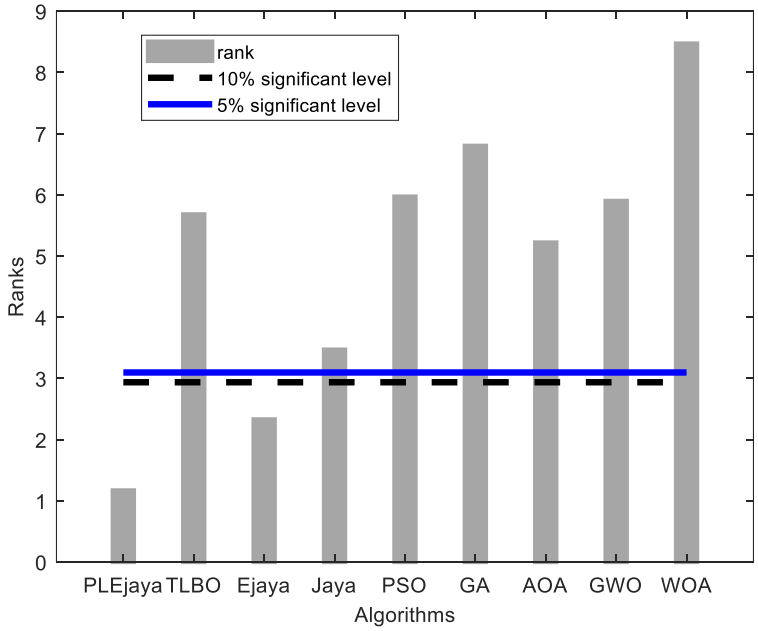

Figure 9. Bar chart presentation of Friedman ranks and Bonferroni-Dunn test for CEC17 functions.

5. Conclusion

The EJaya algorithm is an enhanced version of Jaya algorithm which improves global exploration of the algorithm. However, it also faces some drawbacks such as stagnation in local minima and slow convergence rate due to the single learning strategy and poor population maintenance. This article introduced an enhanced optimization algorithm that works in three phases: Adaptive enhancement, EJAYA, and linear population reduction. Our tests using fifty-three benchmark functions, including 23 standard and 30 CEC 2017 test suite's benchmark functions and compared with eight established meta-heuristic algorithms such as TLBO, EJAYA, JAYA, PSO, GA, AOA, GWO and WOA. Results showed that this PLEJaya effectively finds the best solutions and performs better than TLBO, EJAYA, JAYA, GA, PSO, AOA, GWO, and WOA. When working with complex functions and several variables, it functioned admirably. In every respect, the PLEJaya approach was a reliable and efficient optimization algorithm, leading to superior results and offering hope as a superior answer to several optimization problems.

Conflict of Interest

The authors confirm that there is no conflict of interest to declare for this publication.

Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. The authors would like to thank the editor and anonymous reviewers for their comments that help improve the quality of this work.

AI Disclosure

The author(s) declare that no assistance is taken from generative AI to write this article.

References

Ahmadi, S.A. (2017). Human behavior-based optimization: a novel metaheuristic approach to solve complex optimization problems. *Neural Computing and Applications*, *28*(S1), 233-244. https://doi.org/10.1007/s00521-016-2334-4.

Askari, Q., Younas, I., & Saeed, M. (2020). Political optimizer: a novel socio-inspired meta-heuristic for global optimization. *Knowledge-Based Systems*, *195*, 105709. https://doi.org/10.1016/j.knosys.2020.105709.

Chopra, N., & Ansari, M.M. (2022). Golden jackal optimization: a novel nature-inspired optimizer for engineering applications. *Expert Systems with Applications*, *198*, 116924. https://doi.org/10.1016/j.eswa.2022.116924.

Dehghani, M., & Trojovský, P. (2021). Teamwork optimization algorithm: a new optimization approach for function minimization/maximization. *Sensors*, *21*(13), 4567. https://doi.org/10.3390/s21134567.

Fogel, D.B. (1998). Artificial intelligence through simulated evolution. In: Fogel, D.B. (ed) *Evolutionary Computation: The Fossil Record* (pp. 227-296). Wiley - IEEE press. https://doi.org/10.1109/9780470544600.ch7.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, *13*(5), 533-549. https://doi.org/10.1016/0305-0548(86)90048-1.

Hashim, F.A., Hussain, K., Houssein, E.H., Mabrouk, M.S., & Al-Atabany, W. (2021). Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems. *Applied Intelligence*, *51*(3), 1531-1551. https://doi.org/10.1007/s10489-020-01893-z.

Hatamlou, A. (2013). Black hole: a new heuristic optimization approach for data clustering. *Information Sciences*, *222*, 175-184. https://doi.org/10.1016/j.ins.2012.08.023.

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review*, *42*(1), 21-57. https://doi.org/10.1007/s10462-012-9328-0.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks* (pp. 1942-1948). IEEE. *Perth, WA, Australia*.

Kiran, M.S. (2015). TSA: tree-seed algorithm for continuous optimization. *Expert Systems with Applications*, *42*(19), 6686-6698. https://doi.org/10.1016/j.eswa.2015.04.055.

Kirkpatrick, S., Gelatt, C.D., & Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671-680. https://doi.org/10.1126/science.220.4598.671.

Koza, J.R. (1994). Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, *4*(2), 87-112. https://doi.org/10.1007/BF00175355.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51-67.

Mirjalili, S., Mirjalili, S.M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, *69*, 46-61.

Mousavirad, S.J., & Ebrahimpour-Komleh, H. (2017). Human mental search: a new population-based metaheuristic optimization algorithm. *Applied Intelligence*, *47*(3), 850-887. https://doi.org/10.1007/s10489-017-0903-6.

Passino, K.M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, *22*(3), 52-67. https://doi.org/10.1109/MCS.2002.1004010.

Połap, D., & Woźniak, M. (2021). Red fox optimization algorithm. *Expert Systems with Applications*, *166*, 114107. https://doi.org/10.1016/j.eswa.2020.114107.

Rao, R.V. (2016). Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of Industrial Engineering Computations*, *7*, 19-34.

Rao, R.V., Savsani, V.J., & Vakharia, D.P. (2011). Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, *43*(3), 303-315. https://doi.org/10.1016/j.cad.2010.12.015.

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: a gravitational search algorithm. *Information Sciences*, *179*(13), 2232-2248. https://doi.org/10.1016/j.ins.2009.03.004.

Shi, Y. (2011). Brain storm optimization algorithm. In: Tan, Y., Shi, Y., Chai, Y., & Wang, G. (eds) *Advances in Swarm Intelligence*. Springer Berlin, Heidelberg, pp. 303-309. https://doi.org/10.1007/978-3-642-21515-5_36.

Storn, R., & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341-359. https://doi.org/10.1023/A:1008202821328.

Talbi, E.G. (2009). *Metaheuristics: from design to implementation*. John Wiley & Sons. ISBN: 9780470496909.

Yang, X.S. & Deb, S. (2009). Cuckoo search via Lévy flights. In *2009 World Congress on Nature & Biologically Inspired Computing* (pp. 210-214). IEEE. Coimbatore, India. https://doi.org/10.1109/NABIC.2009.5393690.

Yang, X.S. (2010). A new metaheuristic bat-inspired algorithm. In: González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., & Krasnogor, N. (eds) *Nature Inspired Cooperative Strategies for Optimization* (Vol. 284, pp. 65-74). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-12538-6_6.

Zhang, Y., Chi, A., & Mirjalili, S. (2021). Enhanced Jaya algorithm: a simple but efficient optimization method for constrained engineering design problems. *Knowledge-Based Systems*, *233*, 107555.