NÖHÜ Müh. Bilim. Derg. / NOHU J. Eng. Sci., 2025; 14(2), 597-606
Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi
Niğde Ömer Halisdemir University Journal of Engineering Sciences
Araştırma makalesi / Research article
www.dergipark.org.tr/tr/pub/ngumuh / www.dergipark.org.tr/en/pub/ngumuh

Competitive hybrid Jaya and β-Hill Climbing algorithm for high-cost optimization problems

Yüksek maliyetli optimizasyon problemleri için rekabetçi hibrit Jaya ve β-Hill Climbing algoritması

Gürcan Yavuz[1], Hatem Dumlu[2*], Yacoub Kadar Hassan[3]

[1,2,3] *Kütahya University, Computer Engineering Department, 43100, Kütahya, Türkiye*

Abstract

Jaya algorithm is a population-based parameter-less optimization algorithm. It is frequently used in high-cost industrial and engineering problems. However, Jaya algorithm gets stuck in local optimum in some problems with constrained conditions. In this paper, we introduce a hybrid variant of Jaya to overcome the problem of getting stuck at the local optimum, the β-Hill Climbing local search algorithm is added to the Jaya algorithm and a hybrid Jaya algorithm (CT-JAYA-BH) is presented. Before deciding to use the β-Hill Climbing algorithm, Jaya was also combined with the Quasi Newton and Nelder-Mead local search algorithms. These variants were tested on the CEC 2015 benchmark set provided by IEEE. According to the results, the Jaya-β-Hill Climbing variant (CT-JAYA-BH) obtained the best results. A parameter analysis of CT-JAYA-BH was also performed to determine at which parameter values this variant achieved the best results. Moreover CT-JAYA-BH was compared with 14 different optimization algorithms using the CEC 2015 benchmark set. According to the results, the proposed CT-JAYA-BH algorithm outperforms the other algorithms with an average rank value of 1.87 in both 10 and 30 dimensions. The results show that CT-JAYA-BH is a highly competitive.

Anahtar kelimeler: Jaya, Local Search, β -Hill Climbing, CEC 2015

Öz

Jaya algoritması, popülasyon tabanlı parametresiz bir optimizasyon algoritmasıdır. Yüksek maliyetli endüstriyel ve mühendislik problemlerinde sıklıkla kullanılmaktadır. Ancak, Jaya algoritması kısıtlı koşullara sahip bazı problemlerde yerel optimumda takılıp kalmaktadır. Bu makalede, yerel optimumda takılma probleminin üstesinden gelmek için Jaya'nın hibrit bir varyantı tanıtılmakta, β -Hill Climbing yerel arama algoritması Jaya algoritmasına eklenmekte ve hibrit bir Jaya algoritması (CT-JAYA-BH) sunulmaktadır. β-Hill Climbing algoritmasını kullanmaya karar vermeden önce Jaya, Quasi Newton ve Nelder-Mead yerel arama algoritmaları ile de birleştirilmiştir. Bu varyantlar IEEE tarafından sağlanan CEC 2015 benchmark seti üzerinde test edilmiştir. Sonuçlara göre, Jaya-β-Hill Climbing varyantı (CT-JAYA-BH) en iyi sonuçları elde etmiştir. Bu varyantın hangi parametre değerlerinde en iyi sonuçları elde ettiğini belirlemek için CT-JAYA-BH'nin bir parametre analizi de yapılmıştır. Ayrıca CT-JAYA-BH, CEC 2015 kıyaslama seti kullanılarak 14 farklı optimizasyon algoritması ile karşılaştırılmıştır. Sonuçlara göre, önerilen CT-JAYA-BH algoritması hem 10 hem de 30 boyutta ortalama 1.87 rank değeri ile diğer algoritmalardan daha iyi performans göstermiştir. CT-JAYA-BH oldukça rekabetçidir.

Keywords: Jaya, Yerel Arama, β-Hill Climbing, CEC 2015

1 Introduction

Optimization is the process of maximizing profit, performance, output and efficiency or minimizing costs and energy consumption in engineering and industry [1, 2]. Optimization algorithms are widely used across various fields to address complex computational challenges. However, some problems have high computational costs and time requirements, making them difficult to solve efficiently [3, 4]. These challenges have driven researchers to develop novel algorithms to improve solution quality and efficiency, forming the primary motivation for this study.

Jaya is an optimization algorithm proposed by Rao and used to solve many problems [5–7]. The Jaya algorithm has been used to solve complex and difficult optimization problems in various fields such as engineering and industry. Pandey compares the Jaya algorithm with other algorithms and demonstrates the computational capabilities of the discussed algorithm in both constrained and unconstrained problems [8]. Achanta and Pamula used JAYA for time response and speed control of the system and compared the results with particle swarm optimization (PSO) [9]. Jian and Weng proposed a JAYA algorithm (LCJAYA) using a logistic chaotic map and used it to determine photovoltaic (PV) cell parameters [10]. Zhang et al. combined the self-adaptive classification learning hybrid JAYA and Rao-1 algorithms and applied them to real-world problems [11]. Zhao et al. proposed a surrogate-assisted JAYA algorithm and used it to solve the CEC 2017 benchmark set [12] .

* Sorumlu yazar / Corresponding author, e-posta / e-mail: haten.dumlu@dpı.edu.tr (H. Dumlu)
Geliş / Received: 20.10.2024 Kabul / Accepted: 05.03.2025 Yayımlanma / Published: 15.04.2025
doi:10.28948/ngumuh.1570577

Nayak et al. proposed a new method for automatic diagnosis of sensorineural hearing loss (SNHL) with magnetic resonance imaging (MRI). The features obtained using fast discrete curvelet transform are extracted with PCA+LDA algorithm and MJaya-ELM hybrid classifier is proposed by integrating mutation into Jaya optimization [13]. Another algorithm used in this study is the β-Hill Climbing local search method. Al-Betar introduced the β-Hill Climbing algorithm to the literature as an improved hill climbing algorithm [14]. Researchers have utilized this algorithm in various studies. For example, Ghosh et al. proposed an adaptive β-Hill Climbing based Binary Sailfish Optimizer for feature selection and applied it to 18 datasets from UCI and compared it with 10 algorithms [15]. Abed Alguni and Alkhateeb combined the Cuckoo Search algorithm with the β-Hill Climbing algorithm to improve its efficiency and stabilize its computational cost. They used it to solve 16 test functions and compared it with various metaheuristics [16]. Ghosh et al. applied 18 standard UCI datasets by combining the Coral Reefs Optimizer algorithm with β-Hill Climbing and applied three classifiers KNN, Random Forest and Naive Bayes [17]. Alomari et al. developed an efficient feature selection method for gene selection by combining a Minimum Redundancy Maximum Relevancy filter approach and a hybrid bat algorithm with β-Hill Climbing [18] .

In this paper, a solution based on the Jaya algorithm is proposed for expensive optimization problems where the working budget is very limited, and an efficient solution is required. The proposed approach is based on the β-Hill Climbing algorithm, a variant of the hill climbing local search method. Before choosing β-Hill Climbing algorithm, two different local search approaches were included in the Jaya algorithm to investigate the best local search method for expensive problems. Several experiments were performed to test the performance of the proposed algorithm. For the experiments, the CEC 2015 expensive benchmark set proposed by IEEE for the evolutionary computing session was used for expensive problem types. The algorithm's results are compared to the results of latest metaheuristic algorithms. The study is organized as follows. Section 1 provides a brief introduction to the study. In section 2, the original Jaya algorithm, β-Hill Climbing algorithm and proposed algorithm are mentioned. Section 3 describes the experiments performed and the results obtained. Section 4 contains the conclusion.

2 Materials and methods

2.1 The original Jaya algorithm

Jaya is a parameter-less, population-based, stochastic optimization algorithm proposed by Rao in 2016 [6, 7, 19]. Jaya, which means victory in Sanskrit language, has an approach that always aims for the best solution. The general outline of the algorithm is given in Algorithm 1.

Jaya algorithm starts the optimization process by generating a population of individuals representing randomly generated solutions. It continues the optimization process by updating the positions of the solutions in the population with Equation 1. This process continues until the computational budget is completed.

$$X_{i,j}^{Ite+1} = X_{i,j}^{Ite} + r1\left(X_{best,j}^{Ite} - \left|X_{i,j}^{Ite}\right|\right) - r2\left(X_{worst,j}^{Ite} - \left|X_{i,j}^{Ite}\right|\right) \quad (1)$$

Here, $X_{i,j}^{Ite}$ represents the dimension j. of the solution i. at iteration Ite. $X_{best,j}^{Ite}$ and $X_{worst,j}^{Ite}$ are the j. dimension of the individuals with the best and worst fitness values in the population so far, respectively. Lastly, two random values from the interval $[0, 1]$ are $r1$ and $r2$.

Algorithm 1: Pseudo Code of Original Jaya

Determine the size of the population (NP)
Generate the population randomly using uniform distribution.
$Ite \leftarrow 1$
while until the termination condition will meet **do**
Find the individuals of the population with the best and worst fitness values
for $i = 1$ *to* NP **do**
Update the solution with Equation 1.
$i \leftarrow i + 1$
end for
$Ite \leftarrow Ite + 1$
end while

2.2 β-Hill Climbing method

One of the most straightforward local search algorithms is hill climbing. This algorithm faces the challenge of becoming trapped in a local optimum, therefore researchers have proposed new variants of this algorithm [16, 20]. One of them is the β-Hill Climbing algorithm, a hill climbing local search variant proposed by Al-Betar in 2017 [14].The pseudocode of the β-Hill Climbing algorithm is given in Algorithm 2.

B-Hill-Climbing algorithm is one of recent local search algorithms. It has two operators. The N-operator in neighborhood navigation signifies the algorithm's exploitation capability, while the β-operator represents its exploration capability**.** The algorithm works by starting with a random solution. Then it iteratively runs N-operator followed by β-operator. The N-operator employs the function $x = improve(N(x))$ along with a random walk.

$$x_i' = x_i \pm rand(0,1) \times bw \quad (2)$$

Here Equation 2, i denotes an index chosen randomly from N dimensions. bw is the bandwidth between x_i and x_i'.The β-operator assigns a randomly chosen value for x_i' or a value chosen from the available x.

$$x_i' = \begin{cases} x_r & rand \le \beta \\ x_i & otherwise \end{cases} \quad (3)$$

Here Equation 3, with probability $\beta \in [0,1]$, we assign to x_i' by choosing between x_r and x_i according to the value of $rand$.

Algorithm 2. β-Hill Climbing Algorithm

$x_i = lb_i + (ub_i - lb_i) \times rand(0,1)$
Find objective value $(f(x))$
$Ite = 0$
while $Ite \leq maxIte$ **do**
  $x' = improve(N(x))$ $\rightarrow N$
  for $i = 1$ *to* N **do**
    if $rand() \leq \beta$ **then**
      $x_i' = lb_i + (ub_i - lb_i) \times rand(0,1)$ $\rightarrow \beta$
    end if
  end for
  if $f(x') \leq f(x)$ **then**
    $x = x'$
  end if
  $Ite = Ite + 1$
end while

2.3 *Proposed method (CT-JAYA-BH)*

The proposed method aims to overcome the problem of the Jaya algorithm getting stuck at the local optimum. To overcome this problem, the Jaya algorithm is hybridized with the β-Hill Climbing local search algorithm. The proposed algorithm is given in Algorithm 3. In the proposed approach, the algorithm applies the local search method to the best solution x_{best} of the population so far. This is performed depending on the f parameter to save run budget. The local search also has a run budget, which is determined by the value ls.

Algorithm 3. Pseudo Code of Proposed Algorithm

Set the population size (NP)
Generate the population randomly using uniform distribution
$Ite = 1$
while until the termination condition will meet **do**
  Find the individuals of the population with the best and worst fitness values
  for $i = 1$ *to* NP **do**
  Update the positions with Equation 1
  $i = i + 1$
end for
$Ite = Ite + 1$
if $mod(Ite, f) == 0$ **then**
  $x_{best} = ApplyLocalSearch(x_{best})$
end if
end while
procedure $ApplyLocalSearch(x_{best})$
Apply $\beta - HillClimbingAlgorithm(x_{best}, ls)$
return x_{best}
end procedure
procedure $\beta - HillClimbingAlgorithm(x_{best}, ls)$
$x_i = x_{best}$
Find objective value $(f(x))$
$Ite = 0$
while $Ite \leq ls$ **do**
  $x' = improve(N(x))$
  for $i = 1$ *to* N **do**
    if $rand() \leq \beta$ **then**
      update $x_i' = lb_i + (ub_i - lb_i) \times rand(0,1)$
    end if
  end for
  if $f(x') \leq f(x)$ **then**
    $x = x'$
  end if
  $Ite = Ite + 1$
end while
end procedure

3 Experiments

In the experiments, the CEC 2015 expensive optimization benchmark set from IEEE is used [21]. This benchmark set includes 15 black box test functions, and a summary of the benchmark set is listed in Table 1. The CEC 2015 benchmark set includes test functions for the Unimodal, Multimodal, Hybrid and Composition type categories.

Table 1. CEC 2015 Expensive benchmark suite

Categories	No	Functions	F_i*
Unimodal Functions	Fnc1	Rotated Bent Cigar	100
	Fnc2	Rotated Discus	200
Simple Multimodal Functions	Fnc3	Shifted and Rotated Weierstrass	300
	Fnc4	Shifted and Rotated Schwefel's	400
	Fnc5	Shifted and Rotated Katsuura Function	500
	Fnc6	Shifted and Rotated HappyCat Function	600
	Fnc7	Shifted and Rotated HGBat	700
	Fnc8	Shifted and Rotated Expanded Griewank's plus Rosenbrock's	800
	Fnc9	Shifted and Rotated Expanded Scaffer's F6 Function	900
Hybrid Functions	Fnc10	Hybrid Function 1 (N=3)	1000
	Fnc11	Hybrid Function 2 (N=4)	1100
	Fnc12	Hybrid Function 3 (N=5)	1200
Composition Functions	Fnc13	Composition Function 1 (N=5)	1300
	Fnc14	Composition Function 2 (N=3)	1400
	Fnc15	Composition Function 3 (N=5)	1500

The test functions included in the CEC 2015 benchmark set are performed for 10 and 30 dimensions. All functions in the experiments were solved independently 20 times by all algorithms. Execution is performed up to the number of function calls $Dim \times 50$. A summary of the knowledge of the experimental environment is given in Table 2.

Table 2. Experimental environment

Specifications	Value
# of dimensions	10, 30
# of functions	15
# of independent runs	20
#of population	10
Running budget	500, 1500 (Dim x 50)
Computer Specifications	32 GB DD4 RAM, AMD Ryzen 5 5600

The suggested CT-JAYA-BH algorithm's parameters are adjusted to $\beta = 0.1$ and $bw = 0.001$. The experimental algorithms' control parameters, excluding the common ones, are sourced from their respective original papers. These specific parameters are detailed in Table 3.

Table 3. Control parameters of algorithms

Algorithm	Parameter	Value
ASO	Depth weight	50
	Multiplier weight	0.2
BA	fmax	2
	fmin	0
	alpha	0.9
	gamma	0.9
	A_max	2
	r0_max	1
BOA	c	0.01
	p	0.8
EPO	M	2
	f	3
	l	2
FA	alpha	0.2
	beta	1
	gamma	1
GSA	G0	100
	alpha	20
HGSO	num_cluster	5
	K	1
	alpha	1
	beta	1
	L1	5.00E-03
	L2	100
	L3	1.00E-02
	theta	298.15
	eps	0.05
	M1	0.1
	M2	0.2
HHO	beta	1.5
PRO	Pmut	0.06
WSA	tau	0.8
	sl	0.035
	phi	0.001
	lambda	0.75

3.1 Comparison with local search methods

This study also aims to hybridize the Jaya algorithm with different local search methods. For this approach, three different local search methods selected from the literature are tried one by one to determine the best one. β-Hill Climbing, a hill climbing variant, is the first to be selected for use in hybridization. The other selected local search algorithms are Nelder-Mead [22] and Quasi Newton [23]. These methods are hybridized with Jaya and the results obtained with the CEC 2015 benchmark set in 10 and 30 dimensions are given in Table 4 and Table 5, respectively.

Jaya-β-Hill Climbing approach ranked first with an average rank value of 1.53 in 10 dimensions, Jaya-Nelder Mead approach ranked second with a value of 1.60 and the Jaya-Quasi Newton approach ranked third with a value of 2.73. Analysis of the 30 dimensions average values shows that Jaya-β-Hill Climbing obtained 1.33, Jaya-Nelder Mead obtained 1.80 and Jaya-Quasi Newton obtained 2.80. According to the results, the β-Hill Climbing local search approach is chosen as the preferred method for hybridization. The Friedman ranking method was used to compare algorithm performance by ranking them for each function.

The best algorithm receives rank 1, the worst N, and the average rank is calculated, which is the mean rank.

Table 4. Comparison of mean value Jaya + Local search on CEC 2015 (dim=10, f=5, ls=10)

Fnc	Jaya+β-HillClimbing	Jaya+nelderMead	Jaya+quasiNewton
Fnc1	**9.2120E+08**	1.0450E+09	1.1930E+09
Fnc2	6.3970E+04	**5.4800E+04**	6.6250E+04
Fnc3	3.0990E+02	**3.0950E+02**	3.1050E+02
Fnc4	**1.5160E+03**	2.0360E+03	2.3470E+03
Fnc5	5.0290E+02	**5.0230E+02**	5.0310E+02
Fnc6	**6.0190E+02**	6.0200E+02	6.0210E+02
Fnc7	**7.1290E+02**	7.1740E+02	7.1850E+02
Fnc8	**9.8030E+02**	1.0360E+03	1.1840E+03
Fnc9	**9.0410E+02**	**9.0410E+02**	9.0420E+02
Fnc10	9.8240E+05	**5.5370E+05**	9.1370E+05
Fnc11	**1.1120E+03**	1.1200E+03	**1.1120E+03**
Fnc12	1.5390E+03	**1.4120E+03**	1.4920E+03
Fnc13	**1.6680E+03**	1.6700E+03	1.6840E+03
Fnc14	1.6140E+03	**1.6130E+03**	1.6160E+03
Fnc15	**1.9800E+03**	1.9980E+03	2.0300E+03
MeanRank	**1.533**	1.600	2.733

Table 5. Comparison of mean value Jaya + Local search on CEC 2015 (dim=30, f=5, ls=10)

Fnc	Jaya+β-HillClimbing	Jaya+nelderMead	Jaya+quasiNewton
Fnc1	**1.2910E+10**	1.9550E+10	2.0300E+10
Fnc2	**1.4960E+05**	1.5930E+05	1.7730E+05
Fnc3	**3.3840E+02**	3.3900E+02	3.4090E+02
Fnc4	**4.4980E+03**	7.1720E+03	8.5620E+03
Fnc5	5.0420E+02	**5.0310E+02**	5.0410E+02
Fnc6	**6.0240E+02**	6.0340E+02	6.0340E+02
Fnc7	**7.3370E+02**	7.6740E+02	7.6710E+02
Fnc8	**1.1890E+06**	3.8450E+06	4.5500E+06
Fnc9	9.1390E+02	**9.1380E+02**	9.1400E+02
Fnc10	2.5910E+07	**2.2130E+07**	4.0130E+07
Fnc11	**1.1870E+03**	1.2280E+03	1.2400E+03
Fnc12	2.5940E+03	**2.4140E+03**	2.8920E+03
Fnc13	**1.8390E+03**	2.0880E+03	2.1240E+03
Fnc14	**1.6980E+03**	1.7130E+03	1.7380E+03
Fnc15	**2.7390E+03**	2.7970E+03	2.8310E+03
MeanRank	**1.333**	1.800	2.800

3.2 Parameter analysis

Jaya algorithm's hybridization with the β-Hill Climbing method is selected and parameter analysis is performed to determine f and ls, two parameters that determine the operation of local search algorithms. The possible values of these two parameters are 5 and 10 for f and 10, 20 and 30 for ls. In the case where f is 5 and ls is 30, the algorithm gets an average ranking value of 1.93 for 10 dimensions and an average ranking value of 1.47 for 30 dimensions, ahead of the other parameter cases. Table 6 shows the results for 10 dimensions and Table 7 shows the results for 30 dimensions.

Table 6. Parameter analysis of proposed algorithm (CT-JAYA-BH) on CEC 2015 (dim=10)

Fnc.	f-5-ls-10	f-5-ls-20	f-5-ls-30	f-10-ls-10	f-10-ls-20	f-10-ls-30
Fnc1	9.2120E+08	**8.7760E+08**	9.8130E+08	1.0490E+09	1.0540E+09	9.8920E+08
Fnc2	6.3970E+04	**5.6420E+04**	6.2810E+04	7.5450E+04	6.2580E+04	7.5090E+04
Fnc3	3.0990E+02	**3.0900E+02**	3.0970E+02	3.1030E+02	3.0980E+02	3.1000E+02
Fnc4	1.5160E+03	1.1850E+03	**1.0650E+03**	1.8810E+03	1.5450E+03	1.4550E+03
Fnc5	5.0290E+02	5.0290E+02	5.0280E+02	5.0310E+02	5.0280E+02	**5.0270E+02**
Fnc6	6.0190E+02	6.0190E+02	**6.0170E+02**	6.0210E+02	6.0210E+02	**6.0170E+02**
Fnc7	7.1290E+02	7.1290E+02	**7.0940E+02**	7.1540E+02	7.1070E+02	7.1210E+02
Fnc8	9.8030E+02	8.7270E+02	**8.6660E+02**	8.7250E+02	1.0030E+03	9.3490E+02
Fnc9	**9.0410E+02**	**9.0410E+02**	**9.0410E+02**	**9.0410E+02**	9.0420E+02	**9.0410E+02**
Fnc10	9.8240E+05	9.4210E+05	8.5590E+05	1.2720E+06	**8.0240E+05**	1.2360E+06
Fnc11	1.1120E+03	1.1110E+03	1.1110E+03	1.1130E+03	**1.1100E+03**	1.1120E+03
Fnc12	1.5390E+03	1.5240E+03	**1.4610E+03**	1.4930E+03	1.4890E+03	1.4910E+03
Fnc13	1.6680E+03	**1.6500E+03**	1.6530E+03	1.6680E+03	1.6670E+03	1.6650E+03
Fnc14	1.6140E+03	**1.6130E+03**	1.6150E+03	1.6150E+03	**1.6130E+03**	1.6140E+03
Fnc15	1.9800E+03	**1.9430E+03**	1.9540E+03	2.0070E+03	1.9660E+03	1.9780E+03
Mean Rank	3.867	2.200	**1.933**	5.000	3.267	3.267

Table 7. Parameter analysis of proposed algorithm (CT-JAYA-BH) on CEC 2015 (dim=30)

Fnc.	f-5-ls-10	f-5-ls-20	f-5-ls-30	f-10-ls-10	f-10-ls-20	f-10-ls-30
Fnc1	1.2910E+10	**9.3510E+09**	6.5130E+09	1.5300E+10	1.4370E+10	9.6500E+09
Fnc2	**1.4960E+05**	1.6170E+05	1.5980E+05	1.6430E+05	1.5590E+05	1.5370E+05
Fnc3	3.3840E+02	3.3660E+02	3.3660E+02	3.3860E+02	3.3710E+02	**3.3640E+02**
Fnc4	4.4980E+03	3.4960E+03	**3.1440E+03**	5.4920E+03	4.5810E+03	4.2460E+03
Fnc5	5.0420E+02	5.0430E+02	**5.0400E+02**	**5.0400E+02**	5.0450E+02	5.0410E+02
Fnc6	6.0240E+02	6.0180E+02	**6.0120E+02**	6.0290E+02	6.0260E+02	6.0210E+02
Fnc7	7.3370E+02	7.2620E+02	**7.1700E+02**	7.4100E+02	7.3390E+02	7.2770E+02
Fnc8	1.1890E+06	4.2740E+05	**3.2160E+05**	2.3560E+06	1.1710E+06	6.5660E+05
Fnc9	9.1390E+02	9.1390E+02	**9.1380E+02**	9.1390E+02	9.1390E+02	**9.1380E+02**
Fnc10	2.5910E+07	2.1670E+07	**1.5780E+07**	2.0420E+07	1.9800E+07	2.2800E+07
Fnc11	1.1870E+03	1.1930E+03	**1.1860E+03**	1.2070E+03	1.2090E+03	1.1870E+03
Fnc12	2.5940E+03	**2.5740E+03**	2.5860E+03	2.6880E+03	2.6290E+03	2.5960E+03
Fnc13	1.8390E+03	1.8110E+03	**1.7720E+03**	1.9320E+03	1.8410E+03	1.8070E+03
Fnc14	1.6980E+03	1.6960E+03	**1.6940E+03**	1.7140E+03	1.7100E+03	1.6950E+03
Fnc15	2.7390E+03	**2.7220E+03**	2.7400E+03	2.7460E+03	2.7460E+03	2.7580E+03
Mean Rank	3.667	2.733	**1.467**	5.067	4.467	2.867

3.3 Comparison with different optimizations algorithms

After parameter analysis, the results of the suggested approach are compared with those of most recent algorithms. The algorithms included in the comparison are Atom Search Optimization (ASO) [24], Harris Hawks Optimization (HHO) [25], Henry Gas Solubility Optimization (HGSO) [26], Poor and Rich Optimization algorithm (PRO) [27], Butterfly Optimization Algorithm (BOA) [28], Emperor Penguin Optimizer (EPO) [29], Weighted Superposition Attraction (WSA) [30], JAYA (JA) [19], Fruit Fly Optimization Algorithm (FOA) [31], Bat algorithm (BA) [32], Firefly Algorithms (FA) [33], Gravitational Search Algorithm (GSA) [34], Coati Optimization Algorithm (COA) [3] and Siberian Tiger Optimization (STO) [35].

The algorithms solved the test functions in the benchmark set for 10 dimensions and 30 dimensions. The population size of all algorithms is chosen to be 10 and 30 to make the experiments fair. The mean value of the algorithms After 20 independent runs is given in Table (8-a), Table (8-b) for 10 dimensions and in Table (9-a), Table (9-b) for 30 dimensions.

Table (8-a) and Table (8-b) shows that the proposed algorithm is ahead of the other 14 algorithms with an average rank value of 1.87 in 10 dimensions. It is followed by the original Jaya algorithm with an average rank of 3.37.

Table (9-a) and Table (9-b) shows that the proposed algorithm continued its success by achieving an average ranking value of 1.87 in 30 dimensions, similar to the 10 dimensions, and ranked first out of 14 algorithms. The second algorithm is PRO algorithm with an average ranking value of 3.27, following the proposed algorithm.

CT-JAYA-BH ranked first by outperforming its competitors in both 10 and 30 dimensions. Additionally, the runtimes of CT-JAYA-BH and its competitors for each function are presented in Table (10-a), Table (10-b) and Table (11-a) and Table (11-b), with the best values highlighted in bold.

Table (8-a). Proposed algorithm(CT-JAYA-BH-f-5-ls-30) comparison based mean value with other algorithms (dim=10)

Algorithms	Fnc1	Fnc2	Fnc3	Fnc4	Fnc5	Fnc6	Fnc7	Fnc8
ASO	4.7120E+10	2.8440E+05	3.4000E+02	7.6420E+03	5.0300E+02	6.0540E+02	8.3360E+02	6.2460E+07
HHO	5.8740E+10	9.5250E+04	3.4460E+02	8.2470E+03	5.0340E+02	6.0600E+02	8.1880E+02	1.3660E+07
HGSO	5.6740E+10	7.6050E+04	3.4330E+02	8.6690E+03	5.0430E+02	6.0580E+02	8.1490E+02	1.2380E+07
PRO	4.9670E+10	**6.0130E+04**	3.3920E+02	7.1090E+03	5.0420E+02	6.0540E+02	7.9410E+02	4.7320E+06
BOA	6.8250E+10	6.7430E+04	3.4750E+02	8.8180E+03	5.0420E+02	6.0640E+02	8.3050E+02	3.0750E+07
EPO	1.1120E+11	5.3760E+07	3.4450E+02	1.0600E+04	5.0420E+02	6.0880E+02	9.7340E+02	5.3930E+08
WSA	1.0560E+11	9.3890E+05	3.4710E+02	8.6980E+03	5.0440E+02	6.0830E+02	9.3960E+02	3.2820E+08
JA	1.7940E+10	1.8280E+05	3.3990E+02	8.2000E+03	5.0460E+02	6.0310E+02	7.5330E+02	2.1750E+06
FOA	7.8370E+10	2.4810E+08	3.5320E+02	1.1580E+04	5.1300E+02	6.0670E+02	8.4660E+02	5.3760E+07
BA	8.6560E+10	1.7380E+05	3.4550E+02	6.2670E+03	**5.0210E+02**	6.0740E+02	8.9480E+02	1.2130E+08
FA	1.1520E+11	7.0810E+07	3.5110E+02	1.0350E+04	5.0610E+02	6.0940E+02	9.8520E+02	7.2540E+08
GSA	6.7900E+10	6.5160E+05	3.4660E+02	8.3040E+03	5.0230E+02	6.0610E+02	8.2800E+02	4.9730E+07
COA	4.8520E+10	8.5800E+04	3.4200E+02	8.4950E+03	5.0430E+02	6.0520E+02	7.9860E+02	6.6190E+06
STO	6.4120E+10	7.2770E+04	3.4350E+02	7.8310E+03	5.0370E+02	6.0600E+02	8.2460E+02	2.1760E+07
Proposed	**6.5130E+09**	1.5980E+05	**3.3660E+02**	**3.1440E+03**	5.0400E+02	**6.0120E+02**	**7.1700E+02**	**3.2160E+05**

Table (8-b). Proposed algorithm(CT-JAYA-BH-f-5-ls-30) comparison based mean value with other algorithms (dim=10)

Algorithms	Fnc9	Fnc10	Fnc11	Fnc12	Fnc13	Fnc14	Fnc15	MeanRank
ASO	9.1410E+02	1.1160E+08	1.5240E+03	3.4550E+03	2.8960E+03	1.7930E+03	2.9010E+03	6.200
HHO	9.1380E+02	1.0450E+08	1.4000E+03	7.9160E+03	3.2040E+03	1.9530E+03	3.0660E+03	6.467
HGSO	9.1380E+02	9.4420E+07	1.4040E+03	8.4470E+03	2.8220E+03	1.8750E+03	3.2120E+03	6.333
PRO	9.1370E+02	2.1280E+07	1.2700E+03	2.8880E+03	2.3500E+03	1.7460E+03	2.9110E+03	3.267
BOA	9.1390E+02	2.1920E+08	1.6990E+03	4.6570E+05	3.4210E+03	2.0730E+03	4.5610E+03	9.333
EPO	9.1420E+02	1.9440E+08	2.8940E+03	3.0130E+06	5.3320E+03	2.1520E+03	5.1720E+03	12.467
WSA	9.1410E+02	4.1410E+08	2.3200E+03	1.3860E+06	4.4190E+03	2.2440E+03	3.9330E+03	12.067
JA	9.1390E+02	3.2020E+07	1.2070E+03	2.7520E+03	2.0320E+03	1.7360E+03	2.7660E+03	3.933
FOA	9.1510E+02	1.0710E+09	2.2900E+03	2.1600E+06	4.7040E+03	2.3590E+03	6.9480E+03	13.333
BA	9.1410E+02	3.4960E+08	1.9500E+03	3.2530E+05	3.5150E+03	2.1520E+03	3.9380E+03	9.533
FA	9.1440E+02	7.1380E+08	3.5140E+03	9.6000E+06	5.1830E+03	2.4820E+03	5.6150E+03	14.400
GSA	9.1400E+02	2.8550E+08	1.6590E+03	5.3800E+05	3.2520E+03	2.0060E+03	4.4900E+03	9.067
COA	9.1380E+02	6.3190E+07	1.3740E+03	3.0450E+03	2.2940E+03	1.8130E+03	2.9920E+03	4.800
STO	**9.1350E+02**	8.1340E+07	1.4030E+03	6.4960E+03	3.0510E+03	1.9460E+03	3.0270E+03	5.800
Proposed	9.1380E+02	**1.5780E+07**	**1.1860E+03**	**2.5860E+03**	**1.7720E+03**	**1.6940E+03**	**2.7400E+03**	**1.867**

Table (9-a). Proposed algorithm (CT-JAYA-BH-f-5-ls-30) comparison based mean value with other algorithms (dim=30)

Algorithms	Fnc1	Fnc2	Fnc3	Fnc4	Fnc5	Fnc6	Fnc7	Fnc8
ASO	4.7120E+10	2.8440E+05	3.4000E+02	7.6420E+03	5.0300E+02	6.0540E+02	8.3360E+02	6.2460E+07
HHO	5.8740E+10	9.5250E+04	3.4460E+02	8.2470E+03	5.0340E+02	6.0600E+02	8.1880E+02	1.3660E+07
HGSO	5.6740E+10	7.6050E+04	3.4330E+02	8.6690E+03	5.0430E+02	6.0580E+02	8.1490E+02	1.2380E+07
PRO	4.9670E+10	**6.0130E+04**	3.3920E+02	7.1090E+03	5.0420E+02	6.0540E+02	7.9410E+02	4.7320E+06
BOA	6.8250E+10	6.7430E+04	3.4750E+02	8.8180E+03	5.0420E+02	6.0640E+02	8.3050E+02	3.0750E+07
EPO	1.1120E+11	5.3760E+07	3.4450E+02	1.0600E+04	5.0420E+02	6.0880E+02	9.7340E+02	5.3930E+08
WSA	1.0560E+11	9.3890E+05	3.4710E+02	8.6980E+03	5.0440E+02	6.0830E+02	9.3960E+02	3.2820E+08
JA	1.7940E+10	1.8280E+05	3.3990E+02	8.2000E+03	5.0460E+02	6.0310E+02	7.5330E+02	2.1750E+06
FOA	7.8370E+10	2.4810E+08	3.5320E+02	1.1580E+04	5.1300E+02	6.0670E+02	8.4660E+02	5.3760E+07
BA	8.6560E+10	1.7380E+05	3.4550E+02	6.2670E+03	**5.0210E+02**	6.0740E+02	8.9480E+02	1.2130E+08
FA	1.1520E+11	7.0810E+07	3.5110E+02	1.0350E+04	5.0610E+02	6.0940E+02	9.8520E+02	7.2540E+08
GSA	6.7900E+10	6.5160E+05	3.4660E+02	8.3040E+03	5.0230E+02	6.0610E+02	8.2800E+02	4.9730E+07
COA	4.8520E+10	8.5800E+04	3.4200E+02	8.4950E+03	5.0430E+02	6.0520E+02	7.9860E+02	6.6190E+06
STO	6.4120E+10	7.2770E+04	3.4350E+02	7.8310E+03	5.0370E+02	6.0600E+02	8.2460E+02	2.1760E+07
Proposed	**6.5130E+09**	1.5980E+05	**3.3660E+02**	**3.1440E+03**	5.0400E+02	**6.0120E+02**	**7.1700E+02**	**3.2160E+05**

Table (9-b). Proposed algorithm (CT-JAYA-BH-f-5-ls-30) comparison based mean value with other algorithms (dim=30)

AlgorithmsFnc9		Fnc10	Fnc11	Fnc12	Fnc13	Fnc14	Fnc15	MeanRank
ASO	9.1410E+02	1.1160E+08	1.5240E+03	3.4550E+03	2.8960E+03	1.7930E+03	2.9010E+03	6.200
HHO	9.1380E+02	1.0450E+08	1.4000E+03	7.9160E+03	3.2040E+03	1.9530E+03	3.0660E+03	6.467
HGSO	9.1380E+02	9.4420E+07	1.4040E+03	8.4470E+03	2.8220E+03	1.8750E+03	3.2120E+03	6.333
PRO	9.1370E+02	2.1280E+07	1.2700E+03	2.8880E+03	2.3500E+03	1.7460E+03	2.9110E+03	3.267
BOA	9.1390E+02	2.1920E+08	1.6990E+03	4.6570E+05	3.4210E+03	2.0730E+03	4.5610E+03	9.333
EPO	9.1420E+02	1.9440E+08	2.8940E+03	3.0130E+06	5.3320E+03	2.1520E+03	5.1720E+03	12.467
WSA	9.1410E+02	4.1410E+08	2.3200E+03	1.3860E+06	4.4190E+03	2.2440E+03	3.9330E+03	12.067
JA	9.1390E+02	3.2020E+07	1.2070E+03	2.7520E+03	2.0320E+03	1.7360E+03	2.7660E+03	3.933
FOA	9.1510E+02	1.0710E+09	2.2900E+03	2.1600E+06	4.7040E+03	2.3590E+03	6.9480E+03	13.333
BA	9.1410E+02	3.4960E+08	1.9500E+03	3.2530E+05	3.5150E+03	2.1520E+03	3.9380E+03	9.533
FA	9.1440E+02	7.1380E+08	3.5140E+03	9.6000E+06	5.1830E+03	2.4820E+03	5.6150E+03	14.400
GSA	9.1400E+02	2.8550E+08	1.6590E+03	5.3800E+05	3.2520E+03	2.0060E+03	4.4900E+03	9.067
COA	9.1380E+02	6.3190E+07	1.3740E+03	3.0450E+03	2.2940E+03	1.8130E+03	2.9920E+03	4.800
STO	**9.1350E+02**	8.1340E+07	1.4030E+03	6.4960E+03	3.0510E+03	1.9460E+03	3.0270E+03	5.800
Proposed	9.1380E+02	**1.5780E+07**	**1.1860E+03**	**2.5860E+03**	**1.7720E+03**	**1.6940E+03**	**2.7400E+03**	**1.867**

Table (10-a). Comparison of the proposed algorithm (CT-JAYA-BH-f-5-ls-30) with other algorithms in terms of running times (dim=10)

Algorithms Fnc1		Fnc2	Fnc3	Fnc4	Fnc5	Fnc6	Fnc7	Fnc8
ASO	9.17600E-02	1.26811E-02	3.93629E-02	1.04766E-02	1.57429E-02	1.19461E-02	1.00487E-02	9.40230E-03
HHO	3.34420E-02	1.19366E-02	4.87387E-02	8.19010E-03	1.81069E-02	1.06316E-02	8.91895E-03	8.15925E-03
HGSO	4.59961E-02	1.25924E-02	4.64545E-02	1.08228E-02	1.93465E-02	1.64179E-02	1.15989E-02	1.19629E-02
PRO	2.25162E-02	9.44755E-03	5.51497E-02	8.37420E-03	1.83951E-02	1.08687E-02	8.37710E-03	8.45795E-03
BOA	**9.57315E-03**	4.46170E-03	2.29480E-02	4.06745E-03	9.25515E-03	3.98665E-03	3.37030E-03	3.64220E-03
EPO	1.19307E-02	4.76455E-03	2.55155E-02	4.04065E-03	9.56040E-03	4.86855E-03	4.41670E-03	3.88265E-03
WSA	1.38205E-02	4.80375E-03	2.28568E-02	4.29975E-03	9.84275E-03	8.43380E-03	3.96955E-03	3.76980E-03
JA	1.36622E-02	4.63750E-03	2.07585E-02	4.05760E-03	9.51410E-03	4.51765E-03	3.65430E-03	4.19945E-03
FOA	9.97520E-03	4.77635E-03	2.22949E-02	3.95185E-03	8.70760E-03	4.70285E-03	3.73615E-03	3.87700E-03
BA	2.40317E-02	8.33165E-03	2.75215E-02	6.69270E-03	1.39737E-02	6.96255E-03	6.86085E-03	6.96095E-03
FA	2.80083E-02	1.48669E-02	8.92856E-02	1.47975E-02	3.52540E-02	1.76979E-02	1.37971E-02	1.43557E-02
GSA	1.62209E-02	6.25745E-03	2.46440E-02	6.36390E-03	1.08235E-02	5.92765E-03	5.55015E-03	5.38895E-03
COA	3.34478E-02	9.28735E-03	5.34537E-02	6.82755E-03	1.97677E-02	8.25510E-03	5.63400E-03	6.42915E-03
STO	3.36520E-02	1.60291E-02	1.11786E-01	1.41632E-02	3.87074E-02	1.29123E-02	1.21337E-02	1.24408E-02
Proposed	1.18209E-02	**3.51275E-03**	**4.48680E-03**	**1.43120E-03**	**2.66880E-03**	**1.13740E-03**	**1.04585E-03**	**1.13895E-03**

Table (10-b). Comparison of the proposed algorithm (CT-JAYA-BH-f-5-ls-30) with other algorithms in terms of running times (dim=10)

AlgorithmsFnc9		Fnc10	Fnc11	Fnc12	Fnc13	Fnc14	Fnc15
ASO	1.03637E-02	1.05770E-02	1.46525E-02	1.07654E-02	1.51125E-02	1.03370E-02	2.76488E-02
HHO	7.68455E-03	7.70825E-03	1.49557E-02	8.01500E-03	8.52655E-03	8.02950E-03	3.63281E-02
HGSO	1.11704E-02	1.18966E-02	1.63717E-02	1.19119E-02	1.27569E-02	1.16270E-02	2.92318E-02
PRO	8.24450E-03	9.84335E-03	1.52479E-02	9.02370E-03	9.18550E-03	9.44675E-03	3.97274E-02
BOA	3.43525E-03	3.98940E-03	1.00187E-02	4.28685E-03	4.06245E-03	3.99320E-03	1.87498E-02
EPO	4.41315E-03	4.14345E-03	6.99700E-03	4.25080E-03	4.56155E-03	4.36380E-03	1.96844E-02
WSA	4.15615E-03	3.66895E-03	7.49885E-03	4.36925E-03	4.16280E-03	4.28780E-03	2.02062E-02
JA	3.89050E-03	3.76680E-03	7.04270E-03	4.29765E-03	5.14655E-03	4.39330E-03	1.90907E-02
FOA	3.60830E-03	3.53810E-03	7.01490E-03	3.84200E-03	3.83550E-03	4.01875E-03	1.91006E-02
BA	7.35745E-03	6.56525E-03	1.12870E-02	8.30270E-03	7.70935E-03	7.44445E-03	2.25158E-02
FA	1.42211E-02	1.40746E-02	3.01273E-02	1.70208E-02	1.57456E-02	1.78047E-02	8.48136E-02
GSA	5.49480E-03	5.60325E-03	9.16815E-03	6.34900E-03	6.47565E-03	6.17320E-03	2.07502E-02
COA	6.80175E-03	6.48360E-03	1.48064E-02	8.51155E-03	7.18760E-03	7.90000E-03	4.65447E-02
STO	1.44380E-02	1.43507E-02	2.84612E-02	1.95959E-02	1.40703E-02	1.52971E-02	9.29073E-02
Proposed	**1.03320E-03**	**1.01495E-03**	**1.68215E-03**	**1.16075E-03**	**1.10140E-03**	**1.16950E-03**	**3.67050E-03**

Table (11-a). Comparison of the proposed algorithm (CT-JAYA-BH-f-5-ls-30) with other algorithms in terms of running times (dim=30)

Algorithms	Fnc1	Fnc2	Fnc3	Fnc4	Fnc5	Fnc6	Fnc7	Fnc8
ASO	2.3745E-01	5.7365E-02	1.7892E-01	4.3519E-02	8.4023E-02	4.3045E-02	4.6933E-02	4.2224E-02
HHO	6.4116E-02	2.9757E-02	2.6694E-01	2.7798E-02	9.6887E-02	2.5818E-02	3.0645E-02	2.6338E-02
HGSO	1.0938E-01	6.8277E-02	2.1208E-01	5.5752E-02	1.1268E-01	5.1214E-02	5.8730E-02	5.6804E-02
PRO	6.5533E-02	4.3362E-02	2.9981E-01	3.3345E-02	1.1211E-01	3.2639E-02	3.2759E-02	3.2174E-02
BOA	2.4277E-02	1.4953E-02	1.4381E-01	1.3546E-02	5.1523E-02	1.0871E-02	1.1816E-02	1.1232E-02
EPO	2.8869E-02	2.0320E-02	1.4748E-01	1.8417E-02	5.4979E-02	1.5953E-02	1.5738E-02	1.6797E-02
WSA	3.2715E-02	1.6485E-02	1.5832E-01	1.5383E-02	5.7284E-02	1.3761E-02	1.7108E-02	1.3283E-02
JA	2.3891E-02	1.7126E-02	1.4374E-01	1.6123E-02	5.2052E-02	1.3293E-02	1.4729E-02	1.3959E-02
FOA	2.4828E-02	1.6255E-02	1.4096E-01	1.5862E-02	5.2075E-02	1.4542E-02	1.5284E-02	1.4579E-02
BA	7.6627E-02	4.1271E-02	1.6503E-01	4.3836E-02	7.9297E-02	4.1802E-02	4.0993E-02	4.0349E-02
FA	8.4104E-02	5.7736E-02	6.0788E-01	5.8778E-02	2.2689E-01	6.0035E-02	5.7464E-02	5.7144E-02
GSA	5.1468E-02	2.9442E-02	1.5182E-01	2.9286E-02	6.2735E-02	2.9930E-02	2.8063E-02	2.4614E-02
COA	4.7695E-02	2.4260E-02	3.4894E-01	2.4249E-02	1.2291E-01	2.2543E-02	2.0152E-02	2.2250E-02
STO	7.6504E-02	4.3976E-02	7.0000E-01	4.6010E-02	2.3988E-01	4.0838E-02	3.9721E-02	4.2687E-02
Proposed	**1.4774E-02**	**4.4430E-03**	**2.5779E-02**	**3.3243E-03**	**1.0321E-02**	**3.1391E-03**	**3.0517E-03**	**3.1228E-03**

Table (11-b). Comparison of the proposed algorithm (CT-JAYA-BH-f-5-ls-30) with other algorithms in terms of running times (dim=30)

Algorithms	Fnc9	Fnc10	Fnc11	Fnc12	Fnc13	Fnc14	Fnc15
ASO	4.2804E-02	4.7705E-02	6.9410E-02	4.2407E-02	4.8398E-02	4.6201E-02	1.7395E-01
HHO	2.5797E-02	2.7710E-02	7.3564E-02	3.1997E-02	3.5156E-02	3.6132E-02	2.8179E-01
HGSO	5.4602E-02	5.7584E-02	8.4486E-02	5.3053E-02	5.8238E-02	5.6853E-02	2.1164E-01
PRO	3.4325E-02	3.2933E-02	8.3755E-02	4.0856E-02	4.0796E-02	4.4362E-02	3.0919E-01
BOA	1.1683E-02	1.1886E-02	3.5823E-02	1.4836E-02	1.5783E-02	1.7415E-02	1.4750E-01
EPO	1.6618E-02	1.7530E-02	4.1657E-02	1.9630E-02	1.9781E-02	2.1653E-02	1.5327E-01
WSA	1.2859E-02	1.4186E-02	4.1544E-02	1.7946E-02	1.8171E-02	1.9289E-02	1.6324E-01
JA	1.5140E-02	1.6295E-02	3.8928E-02	1.8338E-02	1.8748E-02	2.0998E-02	1.5007E-01
FOA	1.5840E-02	1.4129E-02	3.9898E-02	1.8169E-02	1.8835E-02	2.0640E-02	1.4437E-01
BA	4.3880E-02	3.9021E-02	6.7057E-02	4.4200E-02	4.5541E-02	4.3703E-02	1.7341E-01
FA	5.7172E-02	5.7710E-02	1.6994E-01	7.3353E-02	7.5795E-02	7.5990E-02	6.4163E-01
GSA	2.7135E-02	2.6474E-02	5.3670E-02	2.9786E-02	2.9761E-02	2.8165E-02	1.5468E-01
COA	2.3502E-02	2.3454E-02	9.0252E-02	3.3199E-02	3.2469E-02	3.4939E-02	3.6670E-01
STO	5.0089E-02	4.5600E-02	1.7310E-01	6.2420E-02	6.7081E-02	6.8681E-02	7.2966E-01
Proposed	**3.0640E-03**	**2.9348E-03**	**7.3051E-03**	**3.7306E-03**	**3.5017E-03**	**3.9564E-03**	**2.5079E-02**

4 Conclusion and discussion

In this study, a novel hybrid optimization algorithm, termed CT-JAYA-BH, is introduced to address problems characterized by high computational costs and limited running budgets. This proposed algorithm is developed by integrating the original Jaya algorithm with the β-Hill Climbing local search technique to enhance solution accuracy. Additionally, the Jaya algorithm is hybridized with three well-established local search methods: β-Hill Climbing, Nelder-Mead, and Quasi-Newton methods, and their performance is systematically evaluated using the CEC 2015 benchmark set. Among these hybrid approaches, the variant that combines Jaya with β-Hill Climbing demonstrated superior performance, yielding more effective optimization results than the other two hybrid models. To maximize the efficiency of the Jaya-β-Hill Climbing algorithm, a comprehensive parameter analysis was conducted on the CEC 2015 benchmark set. Subsequently, the refined algorithm was compared against 14 different new release optimization techniques using the same benchmark suite. The experimental results indicate that the proposed CT-JAYA-BH algorithm consistently outperformed its competitors across both 10-dimensional and 30-dimensional problem instances, achieving the highest ranking among all evaluated methods.

In future studies, Jaya algorithm can be combined with different local search algorithms and the most suitable local search algorithm for Jaya algorithm can be determined. Moreover CT-JAYA-BH can be applied to recent expensive engineering problems.

Conflict of interest

The authors declare that there is no conflict of interest.

Similarity rate (iThenticate): 11%

References

[1] A. Chaudhary and B. Bhushan, An improved teaching learning based optimization method to enrich the flight control of a helicopter system. Sādhanā, 48, 4, 222, 2023. https://doi.org/10.1007/s12046-023-02271-4

[2] Z. Chen, J. Zou, and W. Wang, Digital twin-oriented collaborative optimization of fuzzy flexible job shop scheduling under multiple uncertainties, Sādhanā. 48, 2, 78, 2023. https://doi.org/10.1007/s12046-023-02133-z

[3] M. Dehghani, Z. Montazeri, E. Trojovská, and P. Trojovský, Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems. Knowledge-Based Systems, 259, 110011, 2023. https://doi.org/10.1016/j.knosys.2022.110011

[4] A.H. Saadat, M. Fateh, and J. Keighobadi, Grey wolf optimization algorithm-based robust neural learning control of passive torque simulators with predetermined performance. Turkish Journal of Electrical Engineering and Computer Sciences, 32, 126–143, 2024. https://doi.org/10.55730/1300-0632.4059

[5] E.H. Houssein, A.G. Gad, and Y.M. Wazery, Jaya Algorithm and Applications: A Comprehensive Review. in: N. Razmjooy, M. Ashourian, and Z. Foroozandeh, Eds. Metaheuristics and Optimization in Computer and Electrical Engineering, Springer International Publishing, pp. 3–24, Cham, 2021.

[6] R. Venkata Rao, Jaya: An Advanced Optimization Algorithm and its Engineering Applications, 2018.

[7] G. Yavuz, Senior Learning JAYA With Powell's Method and Incremental Population Strategy. IEEE Access, 10, 103765–103780, 2022. https://doi.org/10.1109/ACCESS.2022.3210122

[8] H. M. Pandey, Jaya a novel optimization algorithm: What, how and why? 6th International Conference - Cloud System and Big Data Engineering (Confluence), 728-730, Noida, India, 2016, 14-15 January 2016.

[9] R.K. Achanta and V.K. Pamula, DC motor speed control using PID controller tuned by jaya optimization algorithm. 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), 983–987 , Chennai, India, 2017, 21-22 September 2017

[10] X. Jian and Z. Weng, A logistic chaotic JAYA algorithm for parameters identification of photovoltaic cell and module models. Optik, 203, 164041,2020. https://doi.org/10.1109/ACCESS.2022.3210122

[11] Y.-J. Zhang, Y.-F. Wang, L.-W. Tao, Y.-X. Yan, J. Zhao, and Z.-M. Gao, Self-adaptive classification learning hybrid JAYA and Rao-1 algorithm for large-scale numerical and engineering problems. Engineering Applications of Artificial Intelligence, 114, 105069, 2022. https://doi.org/10.1016/j.engappai.2022.105069

[12] F. Zhao, H. Zhang, L. Wang, et al., A surrogate-assisted Jaya algorithm based on optimal directional guidance and historical learning mechanism. Engineering Applications of Artificial Intelligence, 111, 104775, 2022. https://doi.org/10.1016/j.engappai.2022.104775

[13] D.R. Nayak, Y. Zhang, D.S. Das, and S. Panda, MJaya-ELM: A Jaya algorithm with mutation and extreme learning machine based approach for sensorineural hearing loss detection. Applied Soft Computing, 83, 105626, 2019. https://doi.org/10.1016/j.asoc.2019.105626

[14] M.A. Al-Betar, β-Hill climbing: an exploratory local search. Neural Computing and Applications, 28, 1, 153–168, 2017. https://doi.org/10.1007/s00521-016-2328-2

[15] K.K. Ghosh, S. Ahmed, P.K. Singh, Z.W. Geem, and R. Sarkar, Improved Binary Sailfish Optimizer Based on Adaptive β-Hill Climbing for Feature Selection. IEEE Access, 8, 83548–83560, 2020. https://doi.org/10.1109/ACCESS.2020.2991543

[16] B.H. Abed-alguni and F. Alkhateeb, Intelligent hybrid cuckoo search and β-hill climbing algorithm. Journal of King Saud University - Computer and Information Sciences, 32, 2, 159–173, 2020. https://doi.org/10.1016/j.jksuci.2018.05.003

[17] S. Ahmed, K.K. Ghosh, L. Garcia-Hernandez, A. Abraham, and R. Sarkar, Improved coral reefs optimization with adaptive β-hill climbing for feature selection. Neural Computing and Applications, 33, 12, 6467–6486, 2021. https://doi.org/10.1007/s00521-020-05409-1

[18] O.A. Alomari, A.T. Khader, M.A. Al-Betar, and M.A. Awadallah, A novel gene selection method using modified MRMR and hybrid bat-inspired algorithm with β-hill climbing. Applied Intelligence, 48, 11, 4429–4447, 2018. https://doi.org/10.1007/s10489-018-1207-1

[19] R. Rao, Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations,7, 1, 19–34, 2016.

[20] Z.A.A. Alyasseri, A.T. Khader, M.A. Al-Betar, and M.A. Awadallah, Hybridizing β-hill climbing with wavelet transform for denoising ECG signals. Information Sciences, 429, 229–246, 2018. https://doi.org/10.1016/j.ins.2017.11.026

[21] Q. Chen, B. Liu, Q. Zhang, J. Liang, P. Suganthan, and B. Qu, Problem definitions and evaluation criteria for CEC 2015 special session on bound constrained single-objective computationally expensive numerical optimization. Computational Intelligence Laboratory, Zhengzhou, China, Technical Report, Nov 2014.

[22] J.A. Nelder and R. Mead, A Simplex Method for Function Minimization. The Computer Journal, 7, 4, 308–313, 1965. https://doi.org/10.1093/comjnl/7.4.308

[23] D.F. Shanno, Conditioning of Quasi-Newton Methods for Function Minimization. Mathematics of Computation, 24, 111, 647–656, 1970. https://doi.org/10.2307/2004840

[24] W. Zhao, L. Wang, and Z. Zhang, Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. Knowledge-Based Systems, 163, 283–304, 2019. https://doi.org/10.1016/j.knosys.2018.08.030

[25] A.A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, Harris hawks optimization: Algorithm and applications. Future Generation Computer Systems, 97, 849–872, 2019. https://doi.org/10.1016/j.future.2019.02.028

[26] F.A. Hashim, E.H. Houssein, M.S. Mabrouk, W. Al-Atabany, and S. Mirjalili, Henry gas solubility optimization: A novel physics-based algorithm. Future Generation Computer Systems, 101, 646–667, 2019. https://doi.org/10.1016/j.future.2019.07.015

[27] S.H. Samareh Moosavi and V.K. Bardsiri, Poor and rich optimization algorithm: A new human-based and multi populations algorithm. Engineering Applications of Artificial Intelligence, 86, 165–181, 2019. https://doi.org/10.1016/j.future.2019.07.015

[28] S. Arora and S. Singh, Butterfly optimization algorithm: a novel approach for global optimization. Soft Computing, 23, 715–734, 2019. https://doi.org/10.1007/s00500-018-3102-4

[29] G. Dhiman and V. Kumar, Emperor penguin optimizer: A bio-inspired algorithm for engineering problems. Knowledge-Based Systems, 159, 20–50, 2018. https://doi.org/10.1016/j.knosys.2018.06.001

[30] A. Baykasoğlu and Ş. Akpinar, Weighted Superposition Attraction (WSA): A swarm intelligence algorithm for optimization problems – Part 1: Unconstrained optimization. Applied Soft Computing, 56, 520–540, 2017. https://doi.org/10.1016/j.asoc.2015.10.036

[31] W.-T. Pan, A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example. Knowledge-Based Systems, 26, 69–74, 2012. https://doi.org/10.1016/j.knosys.2011.07.001

[32] X. Yang and A. Hossein Gandomi, Bat algorithm: a novel approach for global engineering optimization. Engineering Computations, 29, 5, 464–483, 2012. https://doi.org/10.1108/02644401211235834

[33] X.-S. Yang, Firefly Algorithms for Multimodal Optimization. In: O. Watanabe and T. Zeugmann, Eds. Stochastic Algorithms: Foundations and Applications. Springer, Berlin, Heidelberg, pp. 169–178, 2009.

[34] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, GSA: A Gravitational Search Algorithm. Information Sciences, 179, 13, 2232–2248, 2009. https://doi.org/10.1016/j.ins.2009.03.004

[35] P. Trojovský, M. Dehghani, and P. Hanuš, Siberian Tiger Optimization: A New Bio-Inspired Metaheuristic Algorithm for Solving Engineering Optimization Problems. IEEE Access, 10, 132396–132431, 2022. https://doi.org/10.1109/ACCESS.2022.3229964