Artificial Intelligence Review (2022) 55:2575–2639
https://doi.org/10.1007/s10462-021-10077-1

Solving continuous optimization problems using the ımproved Jaya algorithm (IJaya)

Emine Baş[1]

Accepted: 16 September 2021 / Published online: 26 September 2021

Abstract

Jaya algorithm is one of the heuristic algorithms developed in recent years. The most important difference from other heuristic algorithms is that it updates its position according to its best and worst position. In addition to its simplicity, there is no algorithm-specific parameter. Because of these advantages, it has been preferred by researchers for problem-solving in the literature. In this study, the random walk phase of the original Jaya algorithm is developed and the Improved Jaya Algorithm (IJaya) is proposed. IJaya has been tested for success in eighteen classic benchmark test functions. Although the performance of the original Jaya algorithm has been tested at low dimensions in the literature, its success in large sizes has not been tested. In this study, IJaya's success in 10, 20, 30, 100, 500, and 1000 dimensions was examined. Also, the success of IJaya was tested in different population sizes. It has been proven that IJaya's performance has increased with the tests performed. Test results show that IJaya displays good performance and can be used as an alternative method for constrained optimization. In addition, three different engineering design problems were tested in different population sizes to demonstrate the achievements of Jaya and IJaya. According to the results, IJaya can be used as an optimization algorithm in the literature for continuous optimization and large-scale optimization problems.

Keywords Constrained optimization · Benchmark functions · Jaya · Large-scale dimension

1 Introduction

In optimization problems, heuristic algorithms try to obtain an optimum or near optimum solution by using experimental information (Murty 2003). Optimization methods often deal with hard-to-solve problems with a large number of decision variables, with many local optima, and computationally expensive calculations: in these situations, searching for solutions that minimize or maximize the given objective function as efficiently as possible can become incredibly challenging (Iacca et al. 2021). Researchers in the literature frequently use optimization algorithms to solve real-world problems, and the use of optimization algorithms is becoming more and more popular. There are many metaheuristic

✉ Emine Baş
emineozcan@selcuk.edu.tr

[1] Kulu Vocational School, Selçuk University, 42075 Konya, Turkey

algorithms developed for many engineering problems in the literature (Baş and Ülker 2020a, b, c; Beşkirli 2021). Original versions of these algorithms often fall short of finding optimum values. That's why they are constantly being developed. With the fast development of big data technologies, many real-world optimization problems show an enormous increase in the number of decision variables, and large-scale optimization has become an active research field in the last decade (Ren et al. 2021). The large-scale optimization problem is an optimization problem in which the number of decision variables (that is the problem dimension) exceeds 100 (Li et al. 2021a, b).

There are many large-scale problems in the literature. For example flexible scheduling (Sun et al. 2019), gene array classification (Min et al. 2018), satellite-module layout design (Teng et al. 2010), and overlapping community detection in largescale networks (Gao et al. 2019). These problems can be formulated as large-scale optimization problems. However, it is still difficult to solve large-scale problems with traditional evolutionary algorithms. It is generally accepted that the performance of traditional evolutionary algorithms will deteriorate significantly when dealing with large-scale problems (Ren et al. 2021). The main reason for this is that the solution space of a problem increases exponentially with increasing size, and it is not insignificant for a traditional evolutionary algorithm to adequately explore a large solution area within acceptable computational time (Ren et al. 2021). Large-scale optimization has attracted a large number of researchers and has led the way in the development of many advanced optimization algorithms in recent years. Beşkirli proposed a new Tree Seed Algorithm based on the roulette wheel strategy (R-TSA) (Beşkirli 2021). With this strategy, the trees selected at the seed production phase of TSA were diversified and the locations of the seeds were updated to prevent it from being stuck in local minima. Li et al. proposed a dynamic sine cosine algorithm (DSCA). The discovery and use of SCA are dynamically balanced. They also introduced a dynamic inertia weight strategy to avoid falling into the local optimum (Li et al. 2021a, b). Deng et al. proposed the ranking-based biased learning swarm optimizer (RBLSO) for large-scale optimization. The proposed RBLSO contains two types of learning strategies, namely, ranking paired learning (RPL) and biased center learning (BCL) (Deng et al. 2019). Deng et al. proposed Quantum differential evolution with a cooperative coevolution framework and hybrid mutation strategy for large-scale optimization. It has combined the quantum computing characteristics of quantum evolutionary algorithm (QEA) and the divide and conquers the idea of cooperative coevolution evolutionary algorithm(CCEA) (Deng et al. 2021). Li et al. proposed an adaptive particle swarm optimizer with decoupled exploration and exploitation for large-scale optimization. Thanks to the novel balancing strategy of exploration and exploitation, the two functions in the proposed algorithm have been independently and simultaneously managed. (Li et al. 2021a, b). Ren et al. proposed An eigenspace divide-and-conquer approach for large-scale optimization (Ren et al. 2021).

Jaya is a population-based algorithm proposed by Rao for solving constrained and unconstrained continuous optimization problems (Rao 2016). Jaya algorithm is a simple and efficient population-based metaheuristic algorithm (Kaveh et al. 2021). In addition to its simplicity, there is no algorithm-specific parameter. The most prominent feature of Jaya is that while updating an individual's position in the search space, it uses both the individual with the highest quality fitness function value and the individual with the worst fitness function value. Thus, Jaya tries to search both in the local search space and the global search space. Also, this allows Jaya to avoid local minimum traps. Despite having these advantages, the Jaya algorithm suffers from some shortcomings, such as undesirable early convergence and the possibility of getting stuck at local minimums due to insufficient population diversity. Because of the advantages of the Jaya, it has attracted the attention of many researchers. Many studies have been done in the literature with the Jaya algorithm. Kaveh et al. proposed an improved Shuffled Jaya algorithm for sizing optimization of skeletal structures with discrete variables. The proposed algorithm uses the concept of the shuffling process to gain superior exploration capability in the search mechanism (Kaveh et al. 2021). Aslan et al. applied the basic Jaya algorithm for binary optimization problems. In binary optimization, the solution space is structured separately for this type of optimization problem. The decision variables of binary optimization problems are the elements of the set [0,1] (Aslan et al. 2019). Iacca et al. proposed an improved Jaya optimization algorithm with Lévy flight. They have investigated by applying Lévy flight to Jaya, a simple yet effective Swarm Intelligence optimization algorithm recently proposed in the literature (Iacca et al. 2021). Chaudhuri and Sahu introduced the Binary Jaya algorithm for the feature selection problem. They proposed a hybrid filter–wrapper approach for the feature selection. The hybrid methods are a combination of good characteristics of filter and wrapper methods (Chaudhuri and Sahu 2021). Warid suggested an original AMTPG-Jaya inspired approach to handle the OPF formulation. Warid proposes the implementation of a recently invented meta-heuristic optimization solver namely, an adaptive multiple teams perturbation-guiding Jaya (AMTPG-Jaya) technique to tackle with diverse single goal optimum power flow (OPF) forms (Warid, 2020). Das et al. introduced a Jaya algorithm-based wrapper method for optimal feature selection in supervised classification (Das et al. xxxx). Caldeira and Gnanavelbabu proposed a discrete Jaya algorithm for multi-objective flexible job-shop scheduling problems (Caldeira and Gnanavelbabu 2021). Nayak et al. introduced a Jaya algorithm with mutation and an extreme learning machine-based approach for sensorineural hearing loss detection (Nayak et al. 2019). Rao and Keesari proposed a multi-team perturbation guiding Jaya algorithm for optimization of wind farm layout (Rao and Keesari 2018). Elattar and ElSayed proposed a Modified JAYA algorithm for optimal power flow incorporating renewable energy sources considering the cost, emission, power loss, and voltage profile improvement (Elattar and ElSayed 2019).

When studies on Jaya are reviewed, discrete, continuous, and binary forms of Jaya have been proposed for various problems. Jaya has attracted the attention of many reviewers because of its simplicity and parameterless nature. When the literature was examined, it was noted that Jaya's performance was not compared for different population sizes at large dimensions for constrained continuous optimization problems. In addition, different techniques have been developed in the literature to improve Jaya (For example xor gate, filter–wrapper approach, etc.). In this study, Jaya's ability to explore the search space is developed. In addition, the application of Jaya to real-world problems (three different engineering design problems) has been carried out and its success has been proven. The success of IJaya has also been demonstrated by statistical tests.

1.1 The main contribution, motivation, and organization

Jaya has been proposed for low dimensional optimization problems and has shown promising results compared to other optimization algorithms. Although the traditional Jaya algorithm can effectively solve several real-world optimization problems, it does not always find the perfect solution. Therefore, the traditional Jaya algorithm is somewhat weak by default. In this paper, the original Jaya algorithm has been developed and an Improved Jaya algorithm is proposed (IJaya). In the original Jaya algorithm, each individual in the population updates their position using the positions of individuals with the best and worst fitness values. In Jaya's random walk stage, an individual tries to approach the position of the best-valued individual while trying to move away from the position of the worst valued individual. Thus, individuals try to approach the most optimal value. The original Jaya has developed its exploitation ability in the search space. This causes a rapid convergence in Jaya. It prevents Jaya from discovering different optimal values in the search space. The original Jaya may not be capable enough of reconnaissance in search space.

In this study, the exploration ability of Jaya was developed by using random individuals in the random walking phase. Jaya has been proposed for constrained and unconstrained optimization problems and it tests on low-scale problems. Real-world problems often have high variables. As the number of dimensions increases, it becomes difficult to solve the problems. As the large-scale optimization problems are solved, the search space of Jaya will be greatly expanded. This will lead to a reduction in the probability of generating first individuals close to the optimal position. Thus, the performance of Jaya drops sharply. The exploration ability of the proposed Jaya was developed by using random individuals in the random walking phase. In this paper, IJaya's performance has been tested not only at low scale dimensions but also at large-scale dimensions. Real-world problems are often large dimensions. Even if an optimization algorithm outperforms at low-scale dimensions, sometimes it may not perform adequately at high dimensions. Therefore, the performance of an optimization algorithm in both low dimensions and high dimensions should be demonstrated in a study. In this study, the performance of the Jaya algorithm in both low and large dimensions has been demonstrated. Jaya's ability to discover different optimal values in the search space has been enhanced. In this study, the ability to explore was developed by applying Jaya to large dimensions for the first time. In addition, the effect of population size on performance was examined. In this paper, the success of Jaya and IJaya in three different population sizes and six different low and large scale dimensions was examined in detail. The performances of Jaya and IJaya have been compared with algorithms frequently used in the literature. The achievements of Jaya and IJaya are also demonstrated by statistical tests. In addition, Jaya and IJaya have also been tested in three different engineering design problems. Thus, the success of Jaya and IJaya in solving real-world problems is also demonstrated in this study. The results are discussed and evaluated.

The organization of the paper is as follows. The original Jaya is studied in Sect. 2, Improved Jaya (IJaya) is detailed in Sect. 3. IJaya and Jaya are tested on eighteen benchmark functions for low and large dimensions in different population sizes in Sect. 4 and obtained results are compared with well-known heuristic methods. The results are evaluated. In addition, Jaya and IJaya have also been tested in three different engineering design problems in this section.

2 The Jaya algorithm

Jaya algorithm is a heuristic algorithm based on swarm intelligence developed by Rao in recent years (Rao 2016). Its name comes from the word ‘victory’ in the Sanskrit language (Iacca et al. 2021). Jaya algorithm has attracted attention with its simple structure. It has as few parameter values as possible. The most distinctive feature of the Jaya algorithm that distinguishes it from other heuristic algorithms is the random walking phase. In the random walking phase, each individual in the population updates their current positions according to the positions of the individuals with the best and worst fitness values of the population. The algorithm always tries to get closer to success (i.e. reaching the best solution) and tries to avoid failure (i.e. moving away from the worst solution) (Rao 2016). Equation 1 shows the random walking stage of the Jaya algorithm (Rao 2016).

```
Step 1. Set the parameters of the problem
Set the dimension of the problem (n)
Set the population size (N)
Set the termination condition (MaxFEs)
Set the lower and upper limits of variables

Step 2. The initialization of the algorithm
Generate N random population solution on the n-dimensional search space
Calculate fitness values of each population individual using the objective function
Determine the best fitness value as the Best
Determine the worst fitness value as the Worst
Set the function evaluation is equal to the number of individuals (FEs=N)

Step 3. Searching process in solution space
 WHILE Fes<=MaxFEs
    FOR Each candidate solution ($X_i$)
           Create a new position using Eq.(1) ($X_i'$)
           Calculate fitness value for $X'_i$
           IF Fitness value $X_i'$ is better than $X_i$
                $X_i = X_i'$;
              Fitness($X_i$)=Fitness($X_i'$) ;
           END IF
           Fes=Fes+1;
     END FOR

Step 4. Find the best and the worst solutions
Set the solution with the best fitness value (Best)
Set the solution with the worst fitness value (Worst)

Step 5. Testing the termination condition
          IF Termination condition is not satisfied
               Go to Step 3
          ELSE
              Report the best solution
          END

END WHILE
```

Fig. 1 The work steps of the Jaya algorithm

$$X'_{i,j} = X_{i,j} + r_1 \times \left(Best_j - \left|X_{i,j}\right|\right) - r_2 \times \left(Worst_j - \left|X_{i,j}\right|\right) \tag{1}$$

where $i = (1, 2, ..., N)$ is candidate solution to be processed on, $j = (1, 2, ..., n)$ is dimensions of the problem, *Best* represents the solution with the best fitness value, *Worst* has represented the solution with the worst fitness value, *r1* and *r2* are a random number in [0,1]. MaxFEs represent the termination condition. The work steps of Jaya is shown in Fig. 1.

3 Improved Jaya algorithm (IJaya)

The Jaya algorithm is a newly developed population-based meta-heuristic proposed by Rao in 2016. Since its structure is simple and offers effective solutions to problems, it has been the working point of many researchers. Also, it was designed in such a way that it needs as few parameters as possible. As a result, it only needs two parameter values (the population size and the maximum number of generations). The main advantage of the Jaya algorithm is the random walk phase. In the random walk phase, Jaya updates the positions of the new candidate solutions by using the positions of the population members with the best and worst fitness values among the population members. Especially Jaya tries to approach the best position in the search space and move away from the worst position. This feature has improved the local search capability for the Jaya algorithm. Thus, it was able to escape from local traps in the search space. However, the Jaya algorithm's search-space exploration ability is not developed. In some cases, the original Jaya algorithm may not be able to find the global optimum, due to the presence of local optima that can get the search trapped (Iacca et al. 2021). Jaya algorithm has the disadvantage of exploring new points in the search space. Therefore, the random walking phase of the Jaya algorithm was updated in this study. Improved Jaya (IJaya) is proposed in the paper. During the movement of candidate solutions, not only the best and worst positions of the population were used, but also the positions of random individuals. In this way, the risk of the algorithm getting stuck at a local optimum is greatly reduced, while still making sufficient local improvements. Jaya offers a balance between exploration and exploitation.

The work steps of IJaya are the same as that of the original Jaya algorithm, shown in Fig. 1. The only difference is that when updating the positions of individuals in Jaya, only Eq. 1 is used, whereas, in IJaya, both Eqs. 1 and 2 are used. How to use Eqs. 1 or 2 in IJaya is determined by Eq. 3.

Although a simple change in the algorithm, this new random walk process led to an increase in overall performance in terms of solution quality, as demonstrated in the experimental section. The biggest reason for this is that there are jumps to different local minimum values in the search space instead of just getting stuck at local minimums. Real-world problems are often large in scale. As the size of the search space increases, the importance of the exploration capabilities of optimization algorithms has also increased. Performing

Table 1 Parameters setup for original Jaya and IJaya

Algorithms	Population size (N)	Dimension(n)	Maximum iteration
Jaya	10, 25, 50	{10, 20, 30, 100, 500, 1000}	500
IJaya	10, 25, 50	{10, 20, 30, 100, 500, 1000}	500

a random walk through the positions of random individuals at some iterations in IJaya allowed IJaya to evaluate local minimums across all search space.

$$Y'_{i,j} = X_{i,j} + r_3 \times \left(X_{i,j} - \left| X_{r,j} \right| \right) \tag{2}$$

where $i = (1, 2, ..., N)$ is candidate solution to be processed on, $j = (1, 2, ..., n)$ is dimensions of problem, r is represents the solution with the random fitness value, r_3 is random number in [0,1].

$$Random\ \ walking\ (selection) = \begin{cases} X'_{i,j}\ , & rand \geq 0.5 \\ Y'_{i,j}, & otherwise \end{cases} \tag{3}$$

4 Experimental results and analysis

Improved Jaya algorithm (IJaya) and Jaya algorithm have been tested on classical benchmark functions in low, medium, and high dimensions. In addition, the performances of the Jaya and IJaya are analyzed according to their population sizes. The algorithms are run under the same conditions. The population sizes (N) are 10, 25, and 50 while the dimensions (n) of the problem are 10, 20, 30, 100, 500, and 1000. The number of maximum iteration is set at 500. These parameters setup are shown in Table 1. The experiments are executed with a 2.3 GHz CPU and 4 GB RAM. Each application has been run twenty times.

4.1 Benchmark functions

Eighteen different Benchmark functions are used to compare the performance of the proposed algorithm with other algorithms (Surjanovic and Bingham 2019; Suganthan et al. 2005; Beşkirli 2021; Baş and Ülker 2020d, e). These functions are shown in Table 2.

Table 2 Classical benchmark functions

Type	Name	Function	Range	f_{min}
Unimodal	Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$	$X^* = (0,\ldots,0); f(X^*) = 0$
	Schwefel 1.2	$f_2(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_J \right)^2$	$[-100, 100]^n$	$X^* = (0,\ldots,0); f(X^*) = 0$
	Quartic	$f_3(x) = \sum_{i=1}^{n} i x_i^4$	$[-1.28, 1.28]^n$	$X^* = (0,\ldots,0); f(X^*) = 0$
	Powell	$f_4(x) = \sum_{i=1}^{n/4} \left(x_{4i-3} + 10x_{4i-2}\right)^2 + 5\left(x_{4i-1} - x_{4i}\right)^2$ $+ \left(x_{4i-2} - 2x_{4i-1}\right)^4 + 10\left(x_{4i-3} - x_{4i}\right)^4$	$[-4, 5]^n$	$X^* = (0,\ldots,0); f(X^*) = 0$
	Rosenbrock	$f_5(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right]$	$[-30, 30]^n$	$X^* = (1,\ldots,1); f(X^*) = 0$
	Dixon&Price	$f_6(x) = \left(x_1 - 1\right)^2 + \sum_{i=2}^{n} i\left(2x_i^2 - x_{i-1}\right)^2$	$[-10, 10]^n$	$x_i = 2^{-((2\text{^}i-2)/2\text{^}i)}; f(X^*) = 0$
	Sum of Squares	$f_7(x) = \sum_{i=1}^{n} i x_i^2$	$[-10, 10]^n$	$X^* = (0,\ldots,0); f(X^*) = 0$
	Zakharov	$f_8(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 i x_i \right)^2$ $+ \left(\sum_{i=1}^{n} 0.5 i x_i \right)^4$	$[-5, 10]^n$	$X^* = (0,\ldots,0); f(X^*) = 0$
	sum_of_different powers	$f_9(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$	$[-1, 1]^n$	$X^* = (0,\ldots,0); f(X^*) = 0$
	Rotated hyper-Ellipsoid	$f_{10}(x) = \sum_{i=1}^{n} \sum_{j=1}^{i} x_j^2$	$[-65{,}536, 65536]^n$	$X^* = (0,\ldots,0); f(X^*) = 0$

Table 2 (continued)

Type	Name	Function	Range	f_{min}
Multimodal	Michalewicz	$f_{11}(x) = -\sum_{i=1}^{n} sin(x_i) sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$	$[0, \pi]^n$	n = 10: f(X*) = − 9,66,015; n = 20: f(X*) = − 19,6370; n = 30: f(X*) = − 29,6309
	Trid	$f_{12}(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$	$[-n^2, n^2]^n$	i = 1,2,…,n); xi = i(n + 1 − i) f(X*) = − n(n + 4)(n − 1)/6
	Levy	$f_{13}(x) = 0.1\left\{ sin^2(3\pi x_i) + \sum_{i=1}^{n} (x_i - 1)^2 \right.$ $\left. [1 + sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + sin^2(2\pi x_n)] \right\} +$ $\sum_{i=1}^{n} u(x_i, 5, 100, 4); u(x_i, a, k, m) =$ $\begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-10, 10]^n$	X* = (1,…,1); f(X*) = 0
	Schwefel	$f_{14}(x) = \sum_{i=1}^{n} -x_i sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]^n$	X* = (420.9687,…,420.9687) f(X*) = − 418,9829xn
	Rastrigin	$f_{15}(x) = \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^n$	X* = (0,…,0); f(X*) = 0
	Ackley	$f_{16}(x) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) -$ $exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + exp(1)$	$[-32, 32]^n$	X* = (0,…,0); f(X*) = 0
	Griewank	$f_{17(x)} = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^n$	X* = (0,…,0); f(X*) = 0

Table 2 (continued)

Type	Name	Function	Range	f_{min}
	Salmon	$f_{18}(x) = 1 - cos\left(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$	$[-100,100]^n$	$X^* = (0,\ldots,0); f(X^*) = 0$

Table 3 The results of the Jaya and IJaya algorithms for the dimensions of 10, 20, and 30 in unimodal Benchmark functions (Population size = 10)

F	Jaya algorithm			IJaya algorithm		
	n = 10	n = 20	n = 30	n = 10	n = 20	n = 30
f1						
Best	2.09E−14	8.56E−05	1.41E+00	**1.47E−14**	**9.79E−05**	**5.51E−01**
Worst	1.69E−10	2.70E−01	6.52E+01	1.27E−10	1.12E−02	6.82E+00
Mean	1.79E−11	3.53E−02	1.34E+01	**1.69E−11**	**3.30E−03**	**2.43E+00**
Std	4.18E−11	6.60E−02	1.66E+01	**2.75E−11**	**2.92E−03**	**1.60E+00**
Rank	−	−	−	+	+	+
f2						
Best	7.28E+04	4.17E+05	7.16E+05	**4.50E−18**	**9.58E−04**	**5.29E+00**
Worst	3.23E+05	1.03E+06	2.08E+06	9.09E−03	2.77E+02	1.26E+04
Mean	1.47E+05	6.57E+05	1.39E+06	**6.35E−04**	**3.00E+01**	**2.15E+03**
Std	6.11E+04	1.34E+05	3.23E+05	**2.05E−03**	**6.31E+01**	**3.72E+03**
Rank	−	−	−	+	+	+
f3						
Best	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Worst	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Rank	=	=	=	=	=	=
f4						
Best	1.76E−05	8.97E+00	1.33E+00	**1.11E−05**	**2.25E−02**	**7.49E−01**
Worst	1.04E+02	1.31E+02	6.02E+02	1.04E−03	3.71E+00	4.27E+01
Mean	5.18E+00	1.82E+01	7.32E+01	**3.77E−04**	**6.13E−01**	**1.12E+01**
Std	2.26E+01	3.88E+01	1.29E+02	**3.15E−04**	**7.80E−01**	**1.02E+01**
Rank	−	−	−	+	+	+
f5						
Best	1.21E−01	3.12E+01	3.61E+03	**1.51E−02**	**3.96E+00**	**1.84E+02**
Worst	2.22E+01	2.93E+02	7.63E+04	1.61E+01	5.01E+02	6.76E+03
Mean	9.74E+00	1.18E+02	2.86E+04	**4.57E+00**	**7.78E+01**	**2.37E+03**
Std	8.42E+00	7.15E+01	1.92E+04	**5.69E+00**	**1.10E+02**	**1.76E+03**
Rank	−	−	−	+	+	+
f6						
Best	3.70E−02	1.56E+00	1.96E+01	**3.35E−03**	**8.16E−02**	**3.27E+00**
Worst	7.95E+01	1.04E+02	2.21E+03	1.37E+00	2.34E+01	1.54E+02
Mean	1.97E+01	5.10E+01	3.23E+02	**3.91E−01**	**4.98E+00**	**3.97E+01**
Std	2.40E+01	3.73E+01	4.89E+02	**4.27E−01**	**6.00E+00**	**3.84E+01**
Rank	−	−	−	+	+	+
f7						
Best	1.36E−16	1.96E−02	9.72E−02	**1.28E−16**	**2.22E−05**	**7.16E−03**
Worst	1.91E−12	4.00E+02	8.44E+00	1.87E−11	1.49E−03	6.90E−01
Mean	2.49E−13	2.50E+01	2.67E+00	**2.10E−13**	**3.61E−04**	**1.89E−01**
Std	5.28E−13	8.87E+01	2.66E+00	**4.15E−13**	**3.36E−04**	**1.51E−01**
Rank	−	−	−	+	+	+

Table 3 (continued)

F	Jaya algorithm			IJaya algorithm		
	n = 10	n = 20	n = 30	n = 10	n = 20	n = 30
f8						
Best	4.36E+01	2.05E+02	6.69E+02	**2.00E+00**	**4.70E+01**	**1.35E+02**
Worst	3.05E+02	1.20E+06	3.85E+09	1.25E+01	1.26E+02	2.71E+02
Mean	1.69E+02	6.21E+04	2.48E+08	**5.63E+00**	**8.23E+01**	**1.83E+02**
Std	6.71E+01	2.60E+05	8.31E+08	**2.91E+00**	**1.96E+01**	**3.54E+01**
Rank	−	−	−	+	+	+
f9						
Best	**7.08E−40**	1.80E−19	4.21E−12	3.67E−33	**1.36E−19**	**1.67E−13**
Worst	9.90E−32	8.16E−09	3.24E−05	3.32E−24	2.21E−13	1.08E−08
Mean	**1.24E−32**	7.93E−10	6.18E−06	1.68E−25	**2.13E−14**	**9.86E−10**
Std	**2.73E−32**	2.00E−09	1.01E−05	7.23E−25	**4.99E−14**	**2.45E−09**
Rank	+	−	−	−	+	+
f10						
Best	**3.28E−09**	1.19E+03	7.20E+06	3.21E−07	**6.24E+02**	**1.51E+06**
Worst	3.02E−05	2.35E+09	6.88E+08	1.39E−04	2.83E+04	1.02E+08
Mean	**2.33E−06**	1.17E+08	1.07E+08	3.45E−05	**7.92E+03**	**1.54E+07**
Std	**6.48E−06**	5.11E+08	1.48E+08	3.85E−05	**7.83E+03**	**2.26E+07**
Rank	+	−	−	−	+	+

4.2 Comparison of the performances of the IJaya and Jaya for the population size of 10

Jaya and IJaya Algorithms are run in low, medium, and high dimensions ({10, 20, 30, 100, 500, and 1000}) for the population size of 10 and the iteration number of 500. Each application has been run twenty times. The mean, best, worst, standard deviation (Std), and rank of the results are calculated for Jaya and IJaya Algorithms. Tables 3 and 4 show the results obtained when the problem dimension is in low dimensions (10, 20, and 30). Tables 5 and 6 show the results obtained when the problem dimension is in large dimensions (100, 500, and 1000). The best outcome values obtained by the algorithms are indicated in bold. Figure 2 shows the rank performances of Jaya and IJaya in terms of the mean results for the population size of 10 based on dimension sizes.

Tables 3 and 4 show the best, worst, mean, and standard deviation of the fitness values of the Jaya and IJaya algorithms for low dimensions (10, 20, 30). The results show that IJaya algorithm has a high success for unimodal benchmark functions at low dimensions. IJaya showed higher success than the Jaya algorithm in multimodal benchmark functions

Table 4 The results of the Jaya and IJaya algorithms for the dimensions of 10, 20, and 30 in multimodal Benchmark functions (Population size = 10)

F	Jaya algorithm			IJaya algorithm		
	n = 10	n = 20	n = 30	n = 10	n = 20	n = 30
f11						
Best	– 8.65E+00	– 1.67E+01	– 1.27E+01	**– 9.61E+00**	**– 1.70E+01**	**– 1.53E+01**
Worst	– 3.86E+00	– 5.91E+00	– 8.28E+00	– 7.85E+00	– 8.87E+00	– 1.17E+01
Mean	– 6.00E+00	– 8.31E+00	– 9.70E+00	**– 8.69E+00**	**– 1.11E+01**	**– 1.32E+01**
Std	1.53E+00	2.17E+00	1.10E+00	**4.52E–01**	**1.38E+00**	**9.43E–01**
Rank	–	–	–	+	+	+
f12						
Best	– 4.04E+03	– 3.75E+04	– 4.74E+04	**– 5.11E+03**	**– 6.26E+04**	**– 2.64E+05**
Worst	5.92E+03	1.03E+05	9.32E+05	– 2.95E+03	– 4.43E+04	– 5.76E+04
Mean	3.30E+02	4.46E+04	4.20E+05	**– 4.44E+03**	**– 5.32E+04**	**– 1.80E+05**
Std	2.48E+03	4.04E+04	2.57E+05	**4.52E–01**	**5.44E+03**	**5.08E+04**
Rank	–	–	–	+	+	+
f13						
Best	2.48E–10	2.29E+00	9.60E+00	**6.16E–13**	**1.96E–04**	**1.83E+00**
Worst	4.18E+00	1.80E+01	3.06E+01	1.16E+00	4.34E+00	6.18E+00
Mean	1.36E+00	1.07E+01	1.98E+01	**2.17E–01**	**1.91E+00**	**4.34E+00**
Std	1.36E+00	4.60E+00	6.54E+00	**3.04E–01**	**1.16E+00**	**1.46E+00**
Rank	–	–	–	+	+	+
f14						
Best	– 2.37E+03	– 2.24E+03	– 2.62E+03	**– 4.19E+03**	**– 8.37E+03**	**– 1.24E+04**
Worst	1.30E+02	1.81E+02	1.03E+03	– 1.69E+03	– 2.81E+03	– 3.66E+03
Mean	– 8.34E+02	– 9.05E+02	– 9.28E+02	**– 3.31E+03**	**– 5.99E+03**	**– 7.18E+03**
Std	6.57E+02	6.17E+02	9.14E+02	**6.34E+02**	**1.63E+02**	**2.39E+02**
Rank	–	–	–	+	+	+
f15						
Best	7.96E+00	2.47E+01	1.07E+02	**5.17E+00**	**2.39E+01**	**5.75E+01**
Worst	1.03E+02	1.89E+02	3.14E+02	2.96E+01	1.01E+02	1.89E+02
Mean	3.33E+01	9.92E+01	1.94E+02	**1.52E+01**	**6.12E+01**	**1.32E+02**
Std	2.14E+01	4.43E+01	5.50E+01	**7.51E+00**	**2.37E+01**	**4.13E+01**
Rank	–	–	–	+	+	+
f16						
Best	8.01E–07	4.50E–02	5.96E+00	**5.49E–07**	**7.05E–03**	**2.97E–01**
Worst	2.00E+01	2.00E+01	2.03E+01	2.32E+00	5.43E+00	3.97E+00
Mean	1.26E+01	1.71E+01	1.85E+01	**1.98E–01**	**9.09E–01**	**2.15E+00**
Std	9.15E+00	6.61E+00	4.13E+00	**6.04E–01**	**1.37E+00**	**9.79E–01**
Rank	–	–	–	+	+	+
f17						
Best	2.22E–02	2.80E–02	6.76E–01	**2.21E–02**	**8.02E–03**	**5.32E–01**
Worst	1.08E+00	3.34E–01	1.99E+00	5.48E–01	7.03E–01	1.11E+00
Mean	4.63E–01	1.28E–01	1.09E+00	**2.42E–01**	**1.06E–01**	**9.13E–01**
Std	2.96E–01	9.20E–02	2.41E–01	**1.60E–01**	**2.32E–01**	**1.47E–01**
Rank	–	–	–	+	+	+

Table 4 (continued)

F	Jaya algorithm			IJaya algorithm		
	n = 10	n = 20	n = 30	n = 10	n = 20	n = 30
f18						
Best	5.33E+01	7.28E+01	9.58E+01	**9.41E+00**	**2.46E+01**	**3.27E+01**
Worst	6.39E+01	8.53E+01	1.05E+02	2.47E+01	4.45E+01	5.28E+01
Mean	5.84E+01	8.14E+01	1.01E+02	**1.98E+01**	**3.07E+01**	**4.38E+01**
Std	3.30E+00	2.86E+00	2.56E+00	**3.30E+00**	**2.80E+00**	**2.44E+00**
Rank	–	–	–	+	+	+

at low dimensions. The IJaya algorithm shows superior performance at 100% of unimodal benchmark functions for dimension = 20 and 30 and IJaya algorithm has performance at 80% of unimodal benchmark functions for dimension = 10. The IJaya algorithm shows performance at 100% of multimodal benchmark functions for 8 out of 8 functions (f11, f12, f13, f14, f15, f16, f17, and f18).

Tables 5 and 6 show the best, worst, mean, and standard deviation of the fitness values of the Jaya and IJaya algorithms for large dimensions (100, 500, 1000). IJaya shows high success at 100% of unimodal benchmark functions for 10 datasets out of 10 datasets (f1, f2, f3, f4, f5, f6, f7, f8, f9, and f10) for large dimensions. IJaya showed higher success than the Jaya algorithm in multimodal benchmark functions at large dimensions. The IJaya algorithm shows superior performance at roughly 87,5% of multimodal benchmark functions for 7 out of 8 functions (f11, f13, f14, f15, f16, f17, and f18) for dimension = 500 and 1000. The IJaya algorithm has performance at 87,5% of multimodal benchmark functions for 7 out of 8 functions (f12, f13, f14, f15, f16, f17, and f18) for dimension = 100.

When the results are examined in general, the performance of IJaya is higher at lower dimensions. As the dimension size increases, it becomes more difficult to find the optimum solutions. IJaya has continued to be successful despite increasing in dimension size. The biggest reason for this is the ability to explore in the search space. Although Jaya is quite successful in finding local optimums in the search space, it has difficulties in finding global optimum points. Thanks to the newly added random walking phase, different local optimums in the search space are also tested. In unimodal benchmark functions, there is only one local optimum, but in multimodal benchmark functions, there are many local optimums. One of these local optima is the global optimum. Although many of the optimization algorithms show superior performance in unimodal benchmark functions, they cannot perform adequately in multimodal benchmark functions. When the experimental results of the IJaya algorithm were examined, it was observed that it showed superior performance in benchmark functions in both groups. This again proved the success of the IJaya algorithm.

4.3 Comparison of the performances of the IJaya and Jaya for the population size of 25

Jaya and IJaya Algorithms are run in low, medium, and high dimensions ({10, 20, 30, 100, 500, and 1000}) for the population size of 25 and the iteration number of 500. Each application has been run twenty times. The mean, best, worst, standard deviation (Std), and rank

Table 5 The results of the Jaya and IJaya algorithms for the dimensions of 100, 500, and 1000 in unimodal Benchmark functions (Population size = 10)

F	Jaya algorithm			IJaya algorithm		
	n = 100	n = 500	n = 1000	n = 100	n = 500	n = 1000
f1						
Best	2.53E+04	6.81E+05	1.60E+06	**7.63E+03**	**3.75E+05**	**9.22E+05**
Worst	6.33E+04	8.15E+05	1.88E+06	2.02E+04	4.73E+05	1.31E+06
Mean	4.05E+04	7.37E+05	1.75E+06	**1.23E+04**	**4.21E+05**	**1.12E+06**
Std	1.04E+04	3.80E+04	7.05E+04	**3.33E+03**	**3.12E+04**	**7.02E+04**
Rank	−	−	−	+	+	+
f2						
Best	1.47E+07	3.76E+08	1.59E+09	**2.56E+05**	**6.79E+07**	**3.03E+08**
Worst	1.96E+07	4.68E+08	1.77E+09	1.86E+06	1.20E+08	5.90E+08
Mean	1.66E+07	4.13E+08	1.68E+09	**8.02E+05**	**9.18E+07**	**4.16E+08**
Std	1.24E+06	2.14E+07	4.98E+07	**4.03E+05**	**1.39E+07**	**3.99E+07**
Rank	−	−	−	+	+	+
f3						
Best	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Worst	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Rank	=	=	=	=	=	=
f4						
Best	3.91E+03	6.30E+04	1.49E+05	**1.48E+03**	**3.74E+04**	**1.06E+05**
Worst	8.71E+03	7.48E+04	1.63E+05	4.64E+03	5.35E+04	1.30E+05
Mean	6.10E+03	6.91E+04	1.56E+05	**2.46E+03**	**4.74E+04**	**1.18E++05**
Std	1.29E+03	3.12E+03	4.22E+03	**7.56E+02**	**4.22E+03**	**7.31E+03**
Rank	−	−	−	+	+	+
f5						
Best	7.28E+07	1.67E+09	3.88E+09	**6.91E+06**	**6.97E+08**	**1.75E+09**
Worst	2.02E+08	2.27E+09	4.66E+09	3.96E+07	1.22E+09	3.52E+09
Mean	1.19E+08	1.87E+09	4.33E+09	**2.27E+07**	**9.16E+08**	**2.58E+09**
Std	3.09E+07	1.43E+08	1.91E+08	**7.85E+06**	**1.42E+08**	**1.12E+08**
Rank	−	−	−	+	+	+
f6						
Best	1.38E+06	1.49E+08	8.76E+08	**2.25E+05**	**6.83E+07**	**3.65E+08**
Worst	3.74E+06	2.32E+08	1.14E+09	8.28E+05	1.24E+08	6.98E+08
Mean	2.67E+06	2.04E+08	9.85E+08	**4.20E+05**	**9.71E+07**	**5.27E+08**
Std	6.68E+05	1.91E+07	5.94E+07	**1.54E+05**	**1.53E+07**	**5.22E+07**
Rank	−	−	−	+	+	+
f7						
Best	1.11E+04	1.53E+06	7.39E+06	**2.85E+03**	**7.84E+05**	**4.37E+06**
Worst	2.23E+04	1.92E+06	9.37E+06	5.76E+03	1.05E+06	5.79E+06
Mean	1.63E+04	1.75E+06	8.38E+06	**4.26E+03**	**9.24E+05**	**5.15E+06**
Std	2.73E+03	1.10E+05	4.72E+05	**8.69E+02**	**7.83E+04**	**4.06E+05**
Rank	−	−	−	+	+	+

Table 5 (continued)

F	Jaya algorithm			IJaya algorithm		
	n = 100	n = 500	n = 1000	n = 100	n = 500	n = 1000
f8						
Best	7.82E+10	1.40E+20	5.67E+22	**8.29E+02**	**2.47E+18**	**8.25E+21**
Worst	6.00E+14	3.66E+20	1.27E+23	1.33E+03	1.44E+19	1.60E+22
Mean	1.53E+14	2.20E+20	9.64E+22	**1.05E+03**	**6.69E+18**	**1.12E+22**
Std	1.54E+14	6.27E+19	2.27E+22	**1.28E+02**	**3.11E+18**	**2.17E+21**
Rank	−	−	−	+	+	+
f9						
Best	3.20E−04	1.76E−03	9.54E−03	**1.66E−07**	**4.70E−06**	**1.63E−05**
Worst	9.34E−02	1.91E−01	2.40E−01	2.58E−04	6.66E−03	5.22E−03
Mean	1.56E−02	6.13E−02	9.50E−02	**3.06E−05**	**1.15E−03**	**1.17E−03**
Std	2.13E−02	5.33E−02	7.82E−02	**5.95E−05**	**1.63E−03**	**1.38E−03**
Rank	−	−	−	+	+	+
f10						
Best	3.27E+11	6.74E+13	3.26E+14	**1.00E+11**	**3.52E+13**	**1.87E+14**
Worst	9.87E+11	8.34E+13	3.79E+14	2.84E+11	4.93E+13	2.49E+14
Mean	6.13E+11	7.35E+13	3.58E+14	**1.92E+11**	**4.13E+13**	**2.23E+14**
Std	1.55E+11	4.10E+12	1.43E+13	**5.27E+10**	**4.05E+12**	**1.41E+13**
Rank	−	−	−	+	+	+

of the results are calculated for Jaya and IJaya Algorithms. Tables 7 and 8 show the results obtained when the problem dimension is in low dimensions (10, 20, and 30). Tables 9 and 10 show the results obtained when the problem dimension is in large dimensions (100, 500, and 1000). The best outcome values obtained by the algorithms are indicated in bold. Figure 3 shows the rank performances of Jaya and IJaya in terms of the mean results for the population size of 25 based on dimension sizes.

Tables 7 and 8 show the best, worst, mean, and standard deviation of the fitness values of the Jaya and IJaya algorithms for low dimensions (10, 20, 30). IJaya showed superior success at 100% of unimodal benchmark functions at low dimensions. The IJaya algorithm shows superior performance at 100% of multimodal benchmark functions for 8 out of 8 functions (f11, f12, f13, f14, f15, f16, f17, and f18).

Tables 9 and 10 show the best, worst, mean, and standard deviation of the fitness values of the Jaya and IJaya algorithms for large dimensions (100, 500, 1000). IJaya shows high success at roughly 100% of unimodal benchmark functions for 10 datasets out of 10 datasets (f1, f2, f3, f4, f5, f6, f7, f8, f9, and f10) for large dimensions. IJaya showed higher success than the Jaya algorithm in multimodal benchmark functions at large dimensions. The IJaya algorithm shows superior performance at roughly 75% of multimodal benchmark functions for 6 out of 8 functions (f13, f14, f15, f16, f17, and f18).

When the results are examined in low and large dimensions, Jaya's success in unimodal benchmark functions is higher than multimodal benchmark functions and IJaya's success in unimodal benchmark functions is higher than multimodal benchmark functions. This shows that it is easier to find the optimum point in unimodal benchmark functions than in multimodal benchmark functions. On the other hand, IJaya is more successful in multimodal

Table 6 The results of the Jaya and IJaya algorithms for the dimensions of 100, 500, and 1000 in multimodal Benchmark functions (Population size = 10)

F	Jaya algorithm			IJaya algorithm		
	n = 100	n = 500	n = 1000	n = 100	n = 500	n = 1000
f11						
Best	− **9.85E+01**	− 9.85E+01	− 1.91E+02	− 2.87E+01	− **1.01E+02**	− **1.95E+02**
Worst	− 8.96E+01	− 8.96E+01	− 1.70E+02	− 2.36E+01	− 9.05E+01	− 1.71E+02
Mean	− **9.34E+01**	− 9.34E+01	− 1.80E+02	− 2.59E+01	− **9.47E+01**	− **1.80E+02**
Std	2.44E+00	2.44E+00	6.32E+00	**1.37E+00**	**2.40E+00**	**4.03E+00**
Rank	+	−	−	−	+	+
f12						
Best	5.67E+07	**3.33E+11**	**1.28E+13**	**2.65E+07**	3.67E+11	1.36E+13
Worst	3.00E+08	5.58E+11	1.63E+13	1.01E+08	5.75E+11	2.21E+13
Mean	1.61E+08	**4.25E+11**	**1.46E+13**	**5.50E+07**	4.56E+11	1.75E+13
Std	5.80E+07	**4.84E+10**	**1.07E+12**	**2.06E+07**	6.34E+10	2.04E+12
Rank	−	+	+	+	−	−
f13						
Best	1.07E+02	1.59E+03	3.68E+03	**4.12E+01**	**7.01E+02**	**2.00E+03**
Worst	1.98E+02	1.92E+03	4.26E+03	6.14E+01	1.19E+03	3.11E+03
Mean	1.48E+02	1.77E+03	3.93E+03	**4.96E+01**	**9.71E+02**	**2.48E+03**
Std	2.05E+01	1.12E+02	1.67E+02	**5.60E+00**	**1.11E+02**	**3.15E+02**
Rank	−	−	−	+	+	+
f14						
Best	− 4.40E+03	− 9.81E+03	− 7.44E+03	− **1.95E+04**	− **2.79E+04**	− **4.68E+04**
Worst	1.13E+03	3.88E+03	9.44E+03	− 6.33E+03	− 1.23E+04	− 1.81E+04
Mean	− 1.37E+03	− 1.04E+03	4.39E+02	− **1.12E+04**	− **1.93E+04**	− **2.60E+04**
Std	1.60E+03	2.99E+03	5.35E+03	**1.26E+03**	**2.44E+03**	**4.95E+03**
Rank	−	−	−	+	+	+
f15						
Best	7.16E+02	5.80E+03	1.24E+04	**4.34E+02**	**4.49E+03**	**1.05E+04**
Worst	1.24E+03	6.48E+03	1.40E+04	8.96E+02	6.14E+03	1.18E+04
Mean	8.64E+02	6.13E+03	1.31E+04	**6.79E+02**	**4.96E+03**	**1.12E+04**
Std	1.16E+02	1.80E+02	4.22E+02	**1.10E+02**	**1.43E+02**	**3.25E+02**
Rank	−	−	−	+	+	+
f16						
Best	2.00E+01	2.00E+01	2.00E+01	**1.10E+01**	**1.88E+01**	**1.93E+01**
Worst	2.06E+01	2.08E+01	2.08E+01	1.61E+01	1.95E+01	1.99E+01
Mean	2.01E+01	2.03E+01	2.04E+01	**1.43E+01**	**1.92E+01**	**1.96E+01**
Std	1.68E−01	3.48E−01	3.19E−01	**1.17E−01**	**1.98E−01**	**1.75E−01**
Rank	−	−	−	+	+	+
f17						
Best	2.08E+02	5.97E+03	1.45E+04	**7.12E+01**	**2.95E+03**	**8.92E+03**
Worst	5.46E+02	7.37E+03	1.70E+04	1.94E+02	4.35E+03	1.20E+04
Mean	3.84E+02	6.78E+03	1.56E+04	**1.12E+02**	**3.76E+03**	**1.01E+04**
Std	8.63E+01	3.68E+02	5.33E+02	**2.93E+01**	**3.52E+02**	**5.13E+02**
Rank	−	−	−	+	+	+

Table 6 (continued)

F	Jaya algorithm			IJaya algorithm		
	n = 100	n = 500	n = 1000	n = 100	n = 500	n = 1000
f18						
Best	1.78E+02	4.05E+02	5.70E+02	**7.45E+01**	**1.93E+02**	**2.47E+02**
Worst	1.87E+02	4.15E+02	5.82E+02	1.05E+02	2.54E+02	3.74E+02
Mean	1.83E+02	4.10E+02	5.78E+02	**9.39E+01**	**2.28E+02**	**3.37E+02**
Std	2.55E+00	2.36E+00	2.73E+00	**2.50E+00**	**1.55E+00**	**2.47E+00**
Rank	−	−	−	+	+	+

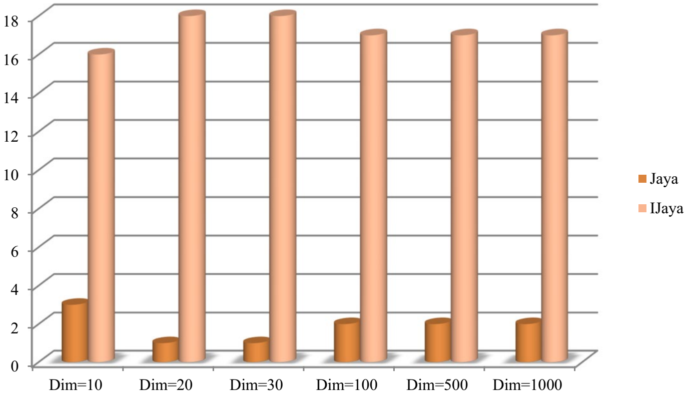


Fig. 2 Rank performances of Jaya and IJaya in terms of the mean results for the population size of 10 based on dimension sizes (Dim = dimension)

benchmark functions than Jaya. This shows that the updated random walk phase is successful. The success of the IJaya algorithm has been proven once again.

4.4 Comparison of the performances of the IJaya and Jaya for the population size of 50

Jaya and IJaya Algorithms are run in low, medium, and high dimensions ({10, 20, 30, 100, 500, and 1000}) for the population size of 50 and the iteration number of 500. Each application has been run twenty times. The mean, best, worst, standard deviation (Std), and rank of the results are calculated for Jaya and IJaya Algorithms. Tables 11 and 12 show the results obtained when the problem dimension is in low dimensions (10, 20, and 30). Tables 13 and 14 show the results obtained when the problem dimension is in large dimensions (100, 500, and 1000). The best outcome values obtained by the algorithms are indicated in bold. Figure 4 shows the rank performances of Jaya and IJaya in terms of the mean results for the population size of 50 based on dimension sizes.

Tables 11 and 12 show the best, worst, mean, and standard deviation of the fitness values of the Jaya and IJaya algorithms for low dimensions (10, 20, 30). The results show that

Table 7 The results of the Jaya and IJaya algorithms for the dimensions of 10, 20, and 30 in unimodal Benchmark functions (Population size = 25)

F	Jaya algorithm			IJaya algorithm		
	n = 10	n = 20	n = 30	n = 10	n = 20	n = 30
f1						
Best	5.95E−07	5.22E−02	1.28E+01	**5.20E−08**	**4.72E−02**	**5.02E+00**
Worst	1.39E−05	3.64E+00	1.28E+02	1.13E−06	3.48E−01	2.27E+01
Mean	3.83E−06	5.67E−01	5.45E+01	**2.51E−07**	**1.22E−01**	**1.35E+01**
Std	2.83E−06	8.45E−01	2.84E+01	**2.34E−07**	**7.45E−02**	**4.33E+00**
Rank	−	−	−	+	+	+
f2						
Best	5.65E+04	3.96E+05	9.65E+05	**8.41E−07**	**4.41E–03**	**1.52E+02**
Worst	2.40E+05	9.92E+05	1.98E+06	6.93E−02	9.89E+00	1.59E+04
Mean	1.56E+05	6.17E+05	1.38E+06	**9.15E−03**	**1.07E+00**	**2.77E+03**
Std	4.60E+04	1.67E+05	3.31E+05	**1.90E−02**	**2.19E+00**	**3.56E+03**
Rank	−	−	−	+	+	+
f3						
Best	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Worst	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Rank	=	=	=	=	=	=
f4						
Best	5.37E−04	1.34E−02	8.69E−01	**2.10E−04**	**1.52E−02**	**1.86E−01**
Worst	1.04E−02	8.54E−01	6.64E+01	3.30E−03	4.55E−01	1.44E+01
Mean	2.98E−03	1.77E−01	7.94E+00	**1.20E−03**	**1.26E−01**	**5.22E+00**
Std	2.70E−03	1.98E−01	1.41E+01	**8.08E−04**	**1.11E−01**	**3.27E+00**
Rank	−	−	−	+	+	+
f5						
Best	1.55E−02	1.52E+01	3.03E+02	**1.50E−02**	**1.09E+01**	**2.28E+02**
Worst	1.49E+02	2.23E+02	4.42E+04	3.01E+01	1.72E+02	5.90E+03
Mean	1.75E+01	9.39E+01	4.76E+03	**4.98E+00**	**6.60E+01**	**2.28E+03**
Std	3.25E+01	5.41E+01	9.82E+03	**7.13E+00**	**4.51E+01**	**1.49E+03**
Rank	−	−	−	+	+	+
f6						
Best	6.21E−01	1.28E+00	1.43E+01	**1.92E−02**	**1.88E−01**	**5.01E+00**
Worst	9.59E+01	8.28E+01	2.22E+02	4.16E−01	7.44E+00	1.01E+02
Mean	2.17E+01	5.13E+01	7.84E+01	**1.60E−01**	**3.26E+00**	**3.24E+01**
Std	2.68E+01	2.29E+01	5.13E+01	**9.41E−02**	**2.15E+00**	**2.14E+01**
Rank	−	−	−	+	+	+
f7						
Best	4.98E−08	1.05E−02	2.03E+00	**3.10E−09**	**4.34E−03**	**6.15E−01**
Worst	1.58E−06	1.63E−01	2.19E+01	4.81E−08	1.71E−02	4.19E+00
Mean	3.77E−07	4.91E−02	7.49E+00	**1.31E−08**	**9.05E−03**	**1.64E+00**
Std	3.93E−07	3.63E−02	4.65E+00	**1.27E−08**	**3.36E−03**	**8.07E−01**
Rank	−	−	−	+	+	+

Table 7 (continued)

F	Jaya algorithm			IJaya algorithm		
	n = 10	n = 20	n = 30	n = 10	n = 20	n = 30
f8						
Best	3.13E+01	2.92E+02	3.83E+02	**6.81E−02**	**3.17E+01**	**1.05E+02**
Worst	2.41E+02	6.77E+02	4.75E+08	9.57E−01	8.53E+01	2.01E+02
Mean	1.04E+02	4.58E+02	2.38E+07	**4.01E−01**	**5.76E+01**	**1.57E+02**
Std	4.89E+01	1.29E+02	1.04E+08	**1.92E−01**	**1.43E+01**	**2.55E+01**
Rank	−	−	−	+	+	+
f9						
Best	7.19E−24	1.20E−16	3.67E−13	**4.40E−24**	**1.27E−16**	**3.54E−13**
Worst	4.21E−20	1.29E−11	1.16E−07	3.93E−21	9.51E−13	1.78E−09
Mean	5.06E−21	7.42E−13	1.12E−08	**6.82E−22**	**1.19E−13**	**2.28E−10**
Std	9.47E−21	2.80E−12	2.74E−08	**9.69E−22**	**2.06E−13**	**3.87E−10**
Rank	−	−	−	+	+	+
f10						
Best	2.06E+00	4.62E+05	7.82E+07	**3.65E−02**	**1.27E+05**	**3.07E+07**
Worst	3.91E+01	9.67E+06	8.93E+08	1.80E+00	9.03E+05	2.17E+08
Mean	1.45E+01	2.40E+06	3.10E+08	**3.91E−01**	**4.51E+05**	**6.79E+07**
Std	9.26E+00	2.13E+06	1.95E+08	**4.04E−01**	**2.10E+05**	**3.84E+07**
Rank	−	−	−	+	+	+

IJaya algorithm has a high success for unimodal benchmark functions at low dimensions. IJaya showed higher success than the Jaya algorithm in multimodal benchmark functions at low dimensions. The IJaya algorithm shows superior performance at 100% of unimodal benchmark functions for dimension = 10, 20, and 30. The IJaya algorithm shows performance at 100% of multimodal benchmark functions for 8 out of 8 functions (f11, f12, f13, f14, f15, f16, f17, and f18) for dimension = 10, 20, and 30.

Tables 13 and 14 show the best, worst, mean, and standard deviation of the fitness values of the Jaya and IJaya algorithms for large dimensions (100, 500, 1000). IJaya shows high success at 100% of unimodal benchmark functions for 10 datasets out of 10 datasets (f1, f2, f3, f4, f5, f6, f7, f8, f9, and f10) for large dimensions. IJaya showed higher success than the Jaya algorithm in multimodal benchmark functions at large dimensions. The IJaya algorithm shows superior performance at roughly 87.5% of multimodal benchmark functions for 7 out of 8 functions (f11, f13, f14, f15, f16, f17, and f18) for dimension = 100. The IJaya algorithm has performance at 75% of multimodal benchmark functions for 6 out of 8 functions (f13, f14, f15, f16, f17, and f18) for dimension = 500 and 1000.

When the results are examined in low and large dimensions, similar results were found in population size 50 as well as in population size 25. The only difference is that IJaya's success in multimodal benchmark functions at large dimensions in population size 50 is more successful than in population size 25. This showed that as the population size increases, the success of IJaya increases even more.

Results show that the eighteen functions achieved very good results for the IJaya in all population sizes for low dimensions (10, 20, 30), while the eighteen functions could not show the same performance for large dimensions (100, 500, 1000). The Jaya algorithm, on the other hand, was observed to achieve a better result in the f11 and f12,

Table 8 The results of the Jaya and IJaya algorithms for the dimensions of 10, 20, and 30 in multimodal Benchmark functions (Population size = 25)

F	Jaya algorithm			IJaya algorithm		
	n = 10	n = 20	n = 30	n = 10	n = 20	n = 30
f11						
Best	− 6.61E+00	− 8.22E+00	− 1.03E+01	**− 8.74E+00**	**− 1.06E+01**	**− 1.39E+01**
Worst	− 3.58E+00	− 5.18E+00	− 6.86E+00	− 6.70E+00	− 8.51E+00	− 1.07E+01
Mean	− 4.95E+00	− 6.73E+00	− 8.87E+00	**− 7.78E+00**	**− 9.41E+00**	**− 1.16E+01**
Std	8.34E−01	8.21E−01	8.90E−01	**5.24E−01**	**5.20E−01**	**7.46E−01**
Rank	−	−	−	+	+	+
f12						
Best	− 4.32E+03	− 6.61E+04	− 2.51E+05	**− 5.14E+03**	**− 6.73E+04**	**− 2.34E+05**
Worst	3.83E+03	8.04E+04	4.94E+05	− 3.64E+03	− 5.52E+03	4.24E+05
Mean	− 1.38E+03	7.86E+03	1.43E+05	**− 4.54E+03**	**− 2.18E+04**	**1.59E+05**
Std	2.47E+03	4.23E+04	2.68E+05	**3.45E+02**	**9.98E+03**	**1.14E+05**
Rank	−	−	−	+	+	+
f13						
Best	1.49E−06	1.25E−01	1.70E+00	**8.71E−09**	**6.62E−01**	**1.59E+00**
Worst	4.54E−01	1.26E+01	2.83E+01	1.22E−06	2.31E+00	1.62E+01
Mean	2.27E−02	2.94E+00	1.23E+01	**1.49E−07**	**1.16E+00**	**1.17E+01**
Std	9.90E−02	3.21E+00	7.37E+00	**2.63E−07**	**4.25E−01**	**2.11E+00**
Rank	−	−	−	+	+	+
f14						
Best	− 1.17E+03	− 4.56E+03	− 3.55E+03	**− 4.19E+03**	**− 8.38E+03**	**− 1.20E+04**
Worst	1.76E+02	− 2.94E+02	1.66E+02	− 1.91E+03	− 2.28E+03	− 2.97E+03
Mean	− 6.90E+02	− 1.16E+03	− 1.06E+03	**− 3.33E+03**	**− 6.80E+03**	**− 8.79E+03**
Std	3.52E+02	8.61E+02	8.06E+02	**3.51E+02**	**8.51E+02**	**8.01E+02**
Rank	−	−	−	+	+	+
f15						
Best	2.53E+01	1.01E+02	1.97E+02	**2.15E+01**	**9.49E+01**	**1.82E+02**
Worst	5.83E+01	1.87E+02	3.17E+02	4.02E+01	1.43E+02	2.49E+02
Mean	3.99E+01	1.46E+02	2.58E+02	**3.01E+01**	**1.21E+02**	**2.24E+02**
Std	8.84E+00	2.52E+01	3.12E+01	**5.09E+00**	**1.21E+01**	**1.68E+01**
Rank	−	−	−	+	+	+
f16						
Best	4.61E−03	1.77E+01	1.92E+01	**1.18E−04**	**1.02E+00**	**5.55E+00**
Worst	1.78E+01	2.00E+01	2.00E+01	8.50E−04	2.90E+00	8.87E+00
Mean	9.19E+00	1.96E+01	1.99E+01	**3.56E−04**	**2.08E+00**	**7.05E+00**
Std	7.78E+00	6.85E−01	1.72E−01	**1.74E−04**	**4.55E−01**	**1.18E−01**
Rank	−	−	−	+	+	+
f17						
Best	2.70E−01	3.82E−01	1.14E+00	**2.00E−01**	**2.30E−01**	**1.07E+00**
Worst	9.03E−01	9.52E−01	2.84E+00	6.50E−01	8.56E−01	1.22E+00
Mean	6.06E−01	7.48E−01	1.64E+00	**5.08E−01**	**5.57E−01**	**1.13E+00**
Std	1.82E−01	1.53E−01	4.97E−01	**1.12E−01**	**1.52E−01**	**3.72E−02**
Rank	−	−	−	+	+	+

Table 8 (continued)

F	Jaya algorithm			IJaya algorithm		
	n = 10	n = 20	n = 30	n = 10	n = 20	n = 30
f18						
Best	8.47E+01	1.27E+02	1.53E+02	**3.32E+01**	**6.17E+01**	**7.36E+01**
Worst	9.53E+01	1.36E+02	1.65E+02	4.84E+01	8.15E+01	1.00E+02
Mean	9.12E+01	1.31E+02	1.59E+02	**4.01E+01**	**6.76E+01**	**8.57E+01**
Std	2.80E+00	2.62E+00	2.96E+00	**2.17E+00**	**2.37E+00**	**2.11E+00**
Rank	−	−	−	+	+	+

especially for the population sizes of 10, 25, and 50. The f3 function was not marked in bold because both algorithms obtained the optimum result. The IJaya algorithm achieved very high success in almost all of the population sizes. It achieved the best results especially for the population sizes of 25 and 50. It was noticed that the Improved Jaya algorithm (IJaya) has obtained a better place than the Jaya algorithm. This is because the exploration capability of the IJaya algorithm is improved in the search space. Search agents are not only using the best or worst values, but also the position information of a random agent of the system during the random walk phase.

Figure 5 shows the performances of Jaya and IJaya algorithms in different population sizes (10, 25, and 50) at low and large scale dimensions (10, 20, 30, 100, 500, and 1000). Four random unimodal and multimodal benchmark functions (f1, f5, f14, and f15) are selected for the convergence graphs. The convergence graphs in Fig. 5 show that the IJaya algorithm converged much faster than the Jaya algorithm.

Figure 6 shows the rank performances of Jaya and IJaya in terms of the mean results in different population sizes 10, 25, and 50 at low and large scale dimensions (10, 20, 30, 100, 500, and 1000). According to the results, as the population size increases, the success of the relevant algorithm increases.

Tables 15, 16, and 17 show the Wilcoxon signed-rank results applied for the results of the Jaya and IJaya algorithms for low and large dimensions. It is set to signed-rank test with the 0.05 *p*-value are given in Tables 15, 16, and 17. The results showed that there is a semantic difference between the results of Jaya and IJaya algorithms. In most cases, it has received a positive sign.

4.5 Comparison of the performances of the IJaya and the other algorithms

Case 1 In this section, alongside the results obtained from the IJaya and Jaya algorithms, literature algorithms such as BA (Yang 2011) and PSO (Xinchao 2010) were used in the solution of eighteen unimodal and multimodal different functions. BA and PSO original codes are taken from the Matlab library (https://www.mathworks.com). The parameter settings of the comparison algorithms are shown in Table 18. The results obtained are compared in Tables 19, 20, 21. All these algorithms were run 20 times and at maximum iteration = 500 and population size = 25 for dimension = 10, 20, 30, 100, 500, and 1000 under the same conditions, and the best, the worst, mean, standard deviation (Std), and rank values were obtained. The best outcome values obtained by the algorithms are indicated in bold.

Table 9 The results of the Jaya and IJaya algorithms for the dimensions of 100, 500, and 1000 in unimodal Benchmark functions (Population size = 25)

F	Jaya algorithm			IJaya algorithm		
	n = 100	n = 500	n = 1000	n = 100	n = 500	n = 1000
f1						
Best	2.65E+04	7.08E+05	1.62E+06	**1.01E+04**	**3.50E+05**	**9.84E+05**
Worst	4.88E+04	8.56E+05	1.82E+06	1.64E+04	4.61E+05	1.17E+06
Mean	3.86E+04	7.73E+05	1.73E+06	**1.32E+04**	**4.09E+05**	**1.06E+06**
Std	5.68E+03	3.37E+04	5.21E+04	**1.82E+03**	**2.73E+04**	**5.00E++04**
Rank	−	−	−	+	+	+
f2						
Best	1.23E+07	3.82E+08	1.59E+09	**9.16E+04**	**3.84E+07**	**3.38E+08**
Worst	1.84E+07	4.47E+08	1.75E+09	1.02E+06	1.10E+08	5.48E+08
Mean	1.59E+07	4.11E+08	1.67E+09	**4.03E+05**	**7.36E+07**	**4.36E+08**
Std	1.56E+06	1.56E+07	4.93E+07	**2.86E+05**	**1.56E+07**	**4.91E+07**
Rank	−	−	−	+	+	+
f3						
Best	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Worst	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Std	0	0	0	0	0	0
Rank	=	=	=	=	=	=
f4						
Best	2.89E+03	6.00E+04	1.39E+05	**1.33E+03**	**3.54E+04**	**8.25E+04**
Worst	5.53E+03	7.11E+04	1.51E+05	2.54E+03	4.77E+04	1.13E+05
Mean	4.40E+03	6.50E+04	1.46E+05	**1.96E+03**	**4.11E+04**	**9.90E+04**
Std	6.97E+02	3.32E+03	3.51E+03	**3.01E+02**	**3.24E+03**	**3.51E+03**
Rank	−	−	−	+	+	+
f5						
Best	5.87E+07	1.48E+09	3.75E+09	**7.94E+06**	**5.45E+08**	**1.40E+09**
Worst	1.56E+08	2.01E+09	4.34E+09	2.26E+07	8.52E+08	2.08E+09
Mean	9.86E+07	1.73E+09	4.03E+09	**1.36E+07**	**6.56E+08**	**1.68E+09**
Std	2.23E+07	1.37E+08	1.43E+08	**3.34E+06**	**8.58E+07**	**1.01E+08**
Rank	−	−	−	+	+	+
f6						
Best	1.00E+06	1.84E+08	8.24E+08	**2.14E+05**	**5.07E+07**	**2.78E+08**
Worst	2.58E+06	2.08E+08	1.01E+09	4.35E+05	8.76E+07	4.48E+08
Mean	1.80E+06	1.94E+08	9.19E+08	**2.97E+05**	**6.80E+07**	**3.67E+08**
Std	4.12E+05	7.68E+06	4.83E+07	**7.04E+04**	**7.57E+06**	**4.45E+07**
Rank	−	−	−	+	+	+
f7						
Best	7.52E+03	1.55E+06	8.07E+06	**4.05E+03**	**7.69E+05**	**4.50E+06**
Worst	2.63E+04	2.00E+06	9.18E+06	6.01E+03	9.55E+05	5.51E+06
Mean	1.57E+04	1.80E+06	8.57E+06	**4.90E+03**	**8.78E+05**	**4.87E+06**
Std	4.20E+03	1.12E+05	3.15E+05	**5.84E+02**	**5.38E+04**	**2.07E+05**
Rank	−	−	−	+	+	+

Table 9 (continued)

F	Jaya algorithm			IJaya algorithm		
	n = 100	n = 500	n = 1000	n = 100	n = 500	n = 1000
f8						
Best	3.52E+12	5.24E+19	3.94E+22	**7.86E+02**	**1.05E+18**	**9.72E+21**
Worst	2.35E+14	2.67E+20	9.98E+22	1.10E+03	1.25E+19	1.45E+22
Mean	4.74E+13	1.86E+20	6.84E+22	**9.18E+02**	**7.49E+18**	**1.20E+22**
Std	5.47E+13	5.18E+19	1.69E+22	**5.06E+00**	**2.52E+18**	**1.14E+21**
Rank	−	−	−	+	+	+
f9						
Best	7.63E−06	2.58E−04	4.52E−04	**3.75E−08**	**2.58E−06**	**6.77E−06**
Worst	1.21E−03	1.57E−02	2.53E−02	4.14E−06	1.58E−04	3.03E−04
Mean	2.80E−04	3.17E−03	5.98E−03	**1.12E−06**	**3.74E−05**	**7.68E−05**
Std	3.43E−04	3.27E−03	5.83E−03	**1.161E−06**	**3.72E−05**	**8.69E−05**
Rank	−	−	−	+	+	+
f10						
Best	3.72E+11	7.09E+13	3.47E+14	**1.52E+11**	**3.35E+13**	**1.92E+14**
Worst	1.04E+12	8.86E+13	3.92E+14	2.75E+11	4.16E+13	2.40E+14
Mean	6.47E+11	7.75E+13	3.68E+14	**2.03E+11**	**3.76E+13**	**2.09E+14**
Std	1.61E+11	4.60E+12	1.22E+13	**2.73E+10**	**2.40E+12**	**1.07E+13**
Rank	−	−	−	+	+	+

According to the comparison results, the IJaya algorithm has shown superior performance. The IJaya algorithm shows performance at 94,44% of benchmark functions for 17 out of 18 functions (except for f18) for dimension = 10, 20, 30, and 100. The IJaya algorithm shows performance at 83,33% of benchmark functions for 15 out of 18 functions (except for f11, f12, and f18) for dimension = 500 and 1000. The results showed that the newly added random walk phase improved the performance of the Jaya algorithm.

Table 22 shows the Wilcoxon signed-rank results applied for the results of the BA and IJaya algorithms for low and large dimensions. Table 23 shows the Wilcoxon signed-rank results applied for the results of the PSO and IJaya algorithms for low and large dimensions. It is set to signed-rank test with the 0.05 *p*-value are given in Tables 22 and 23. The results showed that there is a semantic difference between the results of BA, PSO, and IJaya algorithms. In most cases, it has received a positive sign.

Figure 7 shows the performances of BA, PSO, Jaya, and IJaya algorithms in population size = 25 at low and large scale dimensions (10, 20, 30, 100, 500, and 1000). Four random unimodal and multimodal benchmark functions (f1, f5, f14, and f15) are selected for the convergence graphs. The convergence graphs in Fig. 7 show that the IJaya algorithm converged much faster than the other algorithms. Figure 8 shows the rank performances of Jaya, IJaya, BA, and PSO in terms of the mean results for the population size of 25 based on dimension sizes. The results summarized the success of IJaya.

Table 10 The results of the Jaya and IJaya algorithms for the dimensions of 100, 500, and 1000 in multimodal Benchmark functions (Population size = 25)

F	Jaya algorithm			IJaya algorithm		
	n = 100	n = 500	n = 1000	n = 100	n = 500	n = 1000
f11						
Best	− 2.72E+01	**− 1.04E+02**	**− 1.98E+02**	**− 2.80E+01**	− 9.86E+01	− 1.92E+02
Worst	− 1.98E+01	− 9.13E+01	− 1.68E+02	− 2.34E+01	− 9.28E+01	− 1.76E+02
Mean	− 2.28E+01	**− 9.71E+01**	**− 1.84E+02**	**− 2.54E+01**	− 9.52E+01	− 1.82E+02
Std	1.72E+00	3.73E+00	7.43E+00	**1.13E+00**	**1.88E+00**	**3.18E+00**
Rank	−	+	+	+	−	−
f12						
Best	**− 2.06E+07**	**4.47E+10**	**7.59E+12**	6.24E+05	2.26E+11	8.53E+12
Worst	8.30E+07	3.66E+11	1.19E+13	9.65E+07	3.49E+11	1.18E+13
Mean	3.51E+07	**2.14E+11**	**9.29E+12**	**3.10E+07**	2.69E+11	1.03E+13
Std	3.31E+07	1.02E+11	1.22E+12	**2.20E+07**	**3.20E+10**	**9.05E+11**
Rank	−	+	+	+	−	−
f13						
Best	8.38E+01	1.43E+03	3.58E+03	**3.62E+01**	**6.40E+02**	**1.60E+03**
Worst	1.94E+02	2.02E+03	4.44E+03	5.28E+01	1.05E+03	2.48E+03
Mean	1.34E+02	1.70E+03	4.03E+03	**4.60E+01**	**7.90E+02**	**2.15E+03**
Std	2.45E+01	1.41E+02	2.13E+02	**5.02E+00**	**1.08E+02**	**2.10E+02**
Rank	−	−	−	+	+	+
f14						
Best	− 2.88E+03	− 8.36E+03	− 1.22E+04	**− 2.84E+04**	**− 4.87E+04**	**− 5.41E+04**
Worst	1.29E+03	1.28E+04	9.84E+03	− 5.23E+03	− 1.56E+04	− 1.80E+04
Mean	− 1.50E+03	− 5.45E+02	7.03E+01	**− 1.36E+04**	**− 2.99E+04**	**− 3.19E+04**
Std	1.06E+03	4.62E+03	6.99E+03	**1.03E+03**	**4.60E+03**	**1.02E+03**
Rank	−	−	−	+	+	+
f15						
Best	9.69E+02	6.08E+03	1.28E+04	**8.29E+02**	**5.60E+03**	**1.16E+04**
Worst	1.42E+03	7.37E+03	1.55E+04	1.02E+03	6.03E+03	1.27E+04
Mean	1.21E+03	6.66E+03	1.39E+04	**9.36E+02**	**5.80E+03**	**1.21E+04**
Std	1.04E+02	3.34E+02	5.92E+02	**5.11E+01**	**1.23E+02**	**2.24E+02**
Rank	−	−	−	+	+	+
f16						
Best	2.00E+01	2.00E+01	2.00E+01	**1.16E+01**	**1.85E+01**	**1.91E+01**
Worst	2.04E+01	2.07E+01	2.07E+01	1.37E+01	1.91E+01	1.96E+01
Mean	2.00E+01	2.02E+01	2.02E+01	**1.28E+01**	**1.88E+01**	**1.94E+01**
Std	9.24E−02	3.05E−01	2.61E−01	**5.81E−02**	**1.76E−01**	**1.24E−01**
Rank	−	−	−	+	+	+
f17						
Best	1.76E+02	6.53E+03	1.46E+04	**9.21E+01**	**3.33E+03**	**8.82E+03**
Worst	5.65E+02	7.67E+03	1.68E+04	1.67E+02	3.92E+03	1.04E+04
Mean	3.76E+02	6.96E+03	1.57E+04	**1.20E+02**	**3.65E+03**	**9.69E+03**
Std	9.17E+01	3.23E+02	5.64E+02	**1.84E+01**	**1.75E+02**	**4.31E+02**
Rank	−	−	−	+	+	+

Table 10 (continued)

F	Jaya algorithm			IJaya algorithm		
	n = 100	n = 500	n = 1000	n = 100	n = 500	n = 1000
f18						
Best	2.87E+02	6.43E+02	9.09E+02	**1.48E+02**	**3.52E+02**	**4.90E+02**
Worst	2.95E++02	6.50E+02	9.20E+02	1.76E+02	4.44E+02	5.90E+02
Mean	2.91E+02	6.46E+02	9.14E+02	**1.60E+02**	**3.82E+02**	**5.42E+02**
Std	2.03E+00	1.78E+00	3.06E+00	**2.02E+00**	**1.76E+00**	**2.17E+00**
Rank	−	−	−	+	+	+

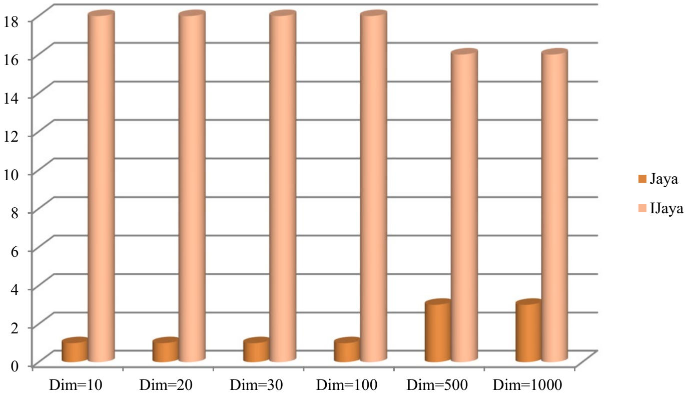


Fig. 3 Rank performances of Jaya and IJaya in terms of the mean results for the population size of 25 based on dimension sizes (Dim = dimension)

Table 11 The results of the Jaya and IJaya algorithms for the dimensions of 10, 20, and 30 in unimodal Benchmark functions (Population size = 50)

F	Jaya algorithm			IJaya algorithm		
	n = 10	n = 20	n = 30	n = 10	n = 20	n = 30
f1						
Best	7.14E−03	1.02E+01	2.76E+02	**2.86E−07**	**6.10E−01**	**3.80E+01**
Worst	4.03E−02	6.16E+01	1.35E+03	1.14E−05	2.28E+00	1.54E+02
Mean	2.47E−02	2.84E+01	7.34E+02	**3.86E−06**	**1.12E+00**	**8.27E+01**
Std	1.06E−02	1.28E+01	2.94E+02	**3.18E−06**	**4.23E−01**	**2.36E+01**
Rank	−	−	−	+	+	+
f2						
Best	6.39E+04	4.67E+05	9.51E+05	**6.10E−01**	**7.74E−07**	**1.12E−02**
Worst	3.17E+05	9.39E+05	1.83E+06	2.28E+00	6.75E+02	6.79E+03
Mean	1.52E+05	6.61E+05	1.38E+06	**1.12E+00**	**4.24E+01**	**5.10E+02**
Std	7.22E+04	1.42E+05	2.71E+05	**4.23E−01**	**1.46E+02**	**1.52E+03**
Rank	−	−	−	+	+	+
f3						
Best	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Worst	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Rank	=	=	=	=	=	=
f4						
Best	2.13E−03	5.89E−01	5.30E+00	**1.03E−04**	**4.96E−01**	**5.10E+00**
Worst	1.39E−02	7.18E+00	9.19E+01	1.65E−03	1.81E+00	2.38E+01
Mean	7.11E−03	2.78E+00	2.77E+01	**8.52E−04**	**9.57E−01**	**1.42E+01**
Std	2.92E−03	1.64E+00	2.22E+01	**4.62E−04**	**3.88E−01**	**4.74E+00**
Rank	−	−	−	+	+	+
f5						
Best	9.21E+00	2.30E+02	5.46E+03	**8.38E−01**	**1.11E+02**	**5.36E+03**
Worst	1.61E+02	1.49E+03	7.31E+04	7.62E+01	3.54E+02	2.28E+04
Mean	4.88E+01	6.77E+02	2.48E+04	**1.19E+01**	**2.28E+02**	**1.35E+04**
Std	4.37E+01	3.56E+02	1.90E+04	**1.57E+01**	**7.96E+01**	**4.85E+03**
Rank	−	−	−	+	+	+
f6						
Best	3.17E+00	4.69E+00	5.10E+01	**6.19E−02**	**1.96E+00**	**4.22E+01**
Worst	7.45E+01	1.16E+02	5.20E+02	1.13E+00	1.41E+01	1.67E+02
Mean	1.65E+01	4.06E+01	2.22E+02	**3.61E−01**	**6.51E+00**	**9.74E+01**
Std	1.79E+01	3.06E+01	1.47E+02	**2.65E−01**	**2.65E+00**	**3.54E+01**
Rank	−	−	−	+	+	+
f7						
Best	5.24E−04	1.16E+00	4.42E+01	**2.86E−08**	**4.08E−02**	**5.60E+00**
Worst	1.83E−03	5.35E+00	1.73E+02	5.68E−07	1.81E−01	1.67E+01
Mean	1.02E−03	2.57E+00	7.71E+01	**1.40E−07**	**1.05E−01**	**8.11E+00**
Std	3.24E−04	1.16E+00	3.13E+01	**1.20E−07**	**3.67E−02**	**2.49E+00**
Rank	−	−	−	+	+	+

Table 11 (continued)

F	Jaya algorithm			IJaya algorithm		
	n = 10	n = 20	n = 30	n = 10	n = 20	n = 30
f8						
Best	3.30E+01	2.46E+02	4.98E+02	**6.89E−01**	**3.55E+01**	**1.10E+02**
Worst	1.57E+02	5.83E+02	9.80E+02	4.15E+00	6.93E+01	1.89E+02
Mean	8.83E+01	4.16E+02	7.66E+02	**1.85E+00**	**5.51E+01**	**1.48E+02**
Std	3.62E+01	9.04E+01	1.31E+02	**8.53E−01**	**9.21E+00**	**2.58E+01**
Rank	−	−	−	+	+	+
f9						
Best	2.10E−14	8.75E−11	7.61E−09	**1.75E−19**	**2.87E−11**	**1.50E−09**
Worst	1.43E−12	7.18E−09	2.90E−06	3.93E−17	8.80E−10	4.81E−08
Mean	3.08E−13	1.49E−09	3.70E−07	**6.89E−18**	**1.95E−10**	**1.05E−08**
Std	3.41E−13	1.62E−09	7.13E−07	**1.03E−17**	**2.14E−10**	**1.23E−08**
Rank	−	−	−	+	+	+
f10						
Best	1.91E+04	5.57E+07	1.77E+09	**1.62E+00**	**1.83E+06**	**2.36E+08**
Worst	9.89E+04	2.65E+08	5.16E+09	1.90E+01	8.78E+06	5.24E+08
Mean	4.25E+04	1.18E+08	3.51E+09	**5.61E+00**	**4.88E+06**	**3.85E+08**
Std	2.15E+04	5.92E+07	1.01E+09	**4.29E+00**	**1.97E+06**	**7.86E+07**
Rank	−	−	−	+	+	+

Case 2 In this section, alongside the results obtained from the IJaya and Jaya algorithms, literature algorithms such as PSO, CSS, GOA, SSA, and MVO were used in the solution of eighteen unimodal and multimodal different functions. The results of PSO, CSS, GOA, SSA, and MVO are taken from (Beşkirli, 2021). The results obtained are compared in Table 24. All these algorithms were run 30 times and at MaxFEs = 500,000 for dimension = 100 under the same conditions and the best, mean, and standard deviation (Std) values were obtained. The best outcome values obtained by the algorithms are indicated in bold.

Comparisons were made for nine benchmark functions (f1, f3, f5, f7, f9, f13, f15, f16, and f17). According to the comparison results, the IJaya algorithm shows performance at 66,67% of benchmark functions for 6 out of 9 functions (except for f9 and f13) for dimension = 100. IJaya algorithm has established a more stable balance between exploration and exploitation with a new random walk phase.

4.6 A comparison of Jaya and IJaya on the engineering design problems

The main purpose of optimization algorithms is to minimize the values of the design parameters and the total cost of the engineering design problem (Rather and Bala 2020). Jaya and IJaya have been applied to three famous mechanical engineering design problems which include the Welded Beam Design problem (WBD), the Compression Spring Design

Table 12 The results of the Jaya and IJaya algorithms for the dimensions of 10, 20, and 30 in multimodal Benchmark functions (Population size = 50)

F	Jaya algorithm			IJaya algorithm		
	n = 10	n = 20	n = 30	n = 10	n = 20	n = 30
f11						
Best	− 5.50E+00	− 7.14E+00	− 1.08E+01	**− 8.13E+00**	**− 1.15E+01**	**− 1.26E+01**
Worst	− 3.72E+00	− 5.37E+00	− 7.36E+00	− 7.12E+00	− 9.07E+00	− 1.08E+01
Mean	− 4.58E+00	− 6.21E+00	− 8.61E+00	**− 7.60E+00**	**− 9.68E+00**	**− 1.16E+01**
Std	4.38E−01	5.04E−01	7.09E−01	**2.94E−01**	**5.01E−01**	**5.19E−01**
Rank	−	−	−	+	+	+
f12						
Best	− 4.77E+03	− 5.59E+04	− 2.52E+05	**− 5.12E+03**	**− 6.25E+04**	**− 2.55E+05**
Worst	4.55E+03	1.04E+05	5.54E+05	− 4.51E+03	− 5.04E+04	− 8.56E+04
Mean	− 4.24E+02	2.48E+04	1.96E+05	**− 4.84E+03**	**− 5.66E+04**	**− 1.85E+05**
Std	2.85E+03	5.07E+04	2.88E+05	**1.60E+02**	**3.11E+03**	**3.92E+04**
Rank	−	−	−	+	+	+
f13						
Best	9.42E−04	3.00E−01	5.32E+00	**1.73E−08**	**7.24E−03**	**2.66E−01**
Worst	8.04E−03	6.02E+00	2.49E+01	6.11E−07	4.54E−02	2.55E+00
Mean	3.62E−03	1.78E+00	1.28E+01	**1.21E−07**	**2.74E−02**	**1.06E+00**
Std	2.03E−03	1.48E+00	5.33E+00	**1.31E−07**	**1.18E−02**	**5.85E−01**
Rank	−	−	−	+	+	+
f14						
Best	− 1.34E+03	− 3.93E+03	− 6.34E+03	**− 4.19E+03**	**− 8.38E+03**	**− 1.26E+04**
Worst	9.53E+01	2.25E+02	− 6.53E+01	− 1.22E+03	− 3.43E+03	− 6.05E+03
Mean	− 7.30E+02	− 9.55E+02	− 1.73E+03	**− 2.94E+03**	**− 7.09E+03**	**− 1.07E+04**
Std	3.85E+02	8.77E+02	1.64E+03	**3.81E+02**	**1.47E+02**	**1.12E+03**
Rank	−	−	−	+	+	+
f15						
Best	2.33E+01	1.11E+02	2.30E+02	**1.33E+01**	**8.77E+01**	**1.72E+02**
Worst	6.60E+01	1.86E+02	3.34E+02	3.23E+01	1.21E+02	2.19E+02
Mean	4.25E+01	1.51E+02	2.80E+02	**2.45E+01**	**1.08E+02**	**1.95E+02**
Std	8.77E+00	2.11E+01	2.93E+01	**5.07E+00**	**8.15E+00**	**1.37E+01**
Rank	−	−	−	+	+	+
f16						
Best	3.84E+00	1.73E+01	1.88E+01	**3.38E−04**	**5.24E−01**	**3.67E+00**
Worst	1.76E+01	2.00E+01	2.00E+01	1.47E−03	1.65E+00	4.30E+00
Mean	1.41E+01	1.96E+01	1.99E+01	**8.71E−04**	**9.30E−01**	**3.96E+00**
Std	4.60E+00	7.17E−01	2.50E−01	**2.46E−04**	**2.64E−01**	**1.70E−01**
Rank	−	−	−	+	+	+
f17						
Best	3.53E−01	1.12E+00	3.54E+00	**1.97E−01**	**7.68E−01**	**1.32E+00**
Worst	7.98E−01	1.62E+00	1.08E+01	5.75E−01	1.03E+00	2.16E+00
Mean	6.05E−01	1.31E+00	6.74E+00	**4.39E−01**	**9.39E−01**	**1.73E+00**
Std	1.11E−01	1.04E−01	2.37E+00	**9.72E−02**	**5.38E−02**	**1.86E−01**
Rank	−	−	−	+	+	+

Table 12 (continued)

F	Jaya algorithm			IJaya algorithm		
	n = 10	n = 20	n = 30	n = 10	n = 20	n = 30
f18						
Best	1.25E+02	1.77E+02	2.21E+02	**4.18E+01**	**6.96E+01**	**8.50E+01**
Worst	1.34E+02	1.88E+02	2.28E+02	6.17E+01	9.94E+01	1.37E+02
Mean	1.30E+02	1.83E+02	2.24E+02	**5.45E+01**	**8.37E+01**	**1.17E+02**
Std	2.42E+00	3.42E+00	2.41E+00	**2.00E+00**	**3.40E+00**	**1.15E+00**
Rank	−	−	−	+	+	+

(CSD), and the Pressure Vessel Design problem (PVD). These problems consist of both equality and inequality constraints. The penalty function method was used to handle the constraints. In the penalty method, if the optimization algorithm violates any constraint, the algorithm is penalized with a high fitness function value (Rather and Bala 2020). The mathematical models of these problems are taking directly from Babalik et al. (2018) and Rather and Bala (2020). In this study, 30,000, 50,000, and 100,000 as the maximum evaluations (MaxFEs) and 20, 40, and 60 as the population sizes were selected, and we tested the success of Jaya and IJaya in three different engineering design problems. 30 independent runs were carried out for each function for all the algorithms. Population size and maximum evaluation parameters were chosen equally to make a fair comparison.

Table 25 shows a comparing Jaya and IJaya algorithms for population size = and 20, 40, and 60 on various engineering design problems for MaxFEs = 30,000. Table 26 shows a comparing Jaya and IJaya algorithms for population size = and 20, 40, and 60 on various engineering design problems for MaxFEs = 50,000. Table 27 shows a comparing Jaya and IJaya algorithms for population size = and 20, 40, and 60 on various engineering design problems for MaxFEs = 100,000. Table 28 shows the results of the Wilcoxon Signed-Rank Test on the results of Jaya and IJaya algorithms on various engineering design problems. Table 29 shows a comparing IJaya and other algorithms on various engineering design problems. Successful results are marked with bold font in Tables 25, 26, 27, and 29.

When the results were examined, the success of Jaya and IJaya increased as the population increased, and in the same way, more optimum results were obtained as the number of MaxFEs increased. According to the results, the results of IJaya are more optimal than Jaya. Thus, the success of IJaya has been proven again in engineering design problems. IJaya has achieved this success by developing not only local search but also global search capability in search space. When IJaya and Jaya are compared with other algorithms in the literature, the results of IJaya and Jaya are satisfactory.

Table 13 The results of the Jaya and IJaya algorithms for the dimensions of 100, 500, and 1000 in unimodal Benchmark functions (Population size = 50)

F	Jaya algorithm			IJaya algorithm		
	n = 100	n = 500	n = 1000	n = 100	n = 500	n = 1000
f1						
Best	4.66E+04	7.69E+05	1.68E+06	**1.74E+04**	**3.68E+05**	**1.03E+06**
Worst	8.01E+04	8.71E+05	1.97E+06	2.56E+04	4.68E+05	1.17E+06
Mean	6.51E+04	8.22E+05	1.82E+06	**2.14E+04**	**4.31E+05**	**1.08E+06**
Std	8.31E+03	2.58E+04	6.23E+04	**2.06E+03**	**2.35E+04**	**4.29E+04**
Rank	−	−	−	+	+	+
f2						
Best	1.27E+07	3.73E+08	1.58E+09	**5.59E+03**	**3.64E+07**	**1.96E+08**
Worst	1.93E+07	4.66E+08	1.75E+09	1.18E+06	9.35E+07	4.64E+08
Mean	1.64E+07	4.16E+08	1.67E+09	**2.68E+05**	**6.04E+07**	**3.14E+08**
Std	1.95E+06	2.05E+07	4.50E+07	**2.85E+05**	**1.68E+07**	**4.49E+07**
Rank	−	−	−	+	+	+
f3						
Best	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Worst	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Rank	=	=	=	=	=	=
f4						
Best	2.81E+03	6.33E+04	1.42E+05	**1.99E+03**	**3.83E+04**	**8.46E+04**
Worst	7.00E+03	6.97E+04	1.54E+05	3.07E+03	4.88E+04	1.11E+05
Mean	4.83E+03	6.60E+04	1.49E+05	**2.52E+03**	**4.29E+04**	**9.64E+04**
Std	1.24E+03	1.79E+03	3.40E+03	**3.10E+02**	**1.21E+03**	**3.10E+03**
Rank	−	−	−	+	+	+
f5						
Best	7.18E+07	1.68E+09	3.81E+09	**1.55E+07**	**5.36E+08**	**1.34E+09**
Worst	1.69E+08	2.09E+09	4.29E+09	2.75E+07	7.51E+08	1.80E+09
Mean	1.17E+08	1.84E+09	4.15E+09	**2.15E+07**	**6.33E+08**	**1.48E+09**
Std	2.74E+07	9.42E+07	1.23E+08	**3.42E+06**	**6.12E+07**	**1.05E+08**
Rank	−	−	−	+	+	+
f6						
Best	1.01E+06	1.87E+08	8.70E+08	**3.27E+05**	**5.53E+07**	**2.79E+08**
Worst	3.12E+06	2.27E+08	9.79E+08	5.63E+05	7.01E+07	4.01E+08
Mean	2.06E+06	2.07E+08	9.25E+08	**4.33E+05**	**6.26E+07**	**3.37E+08**
Std	5.17E+05	1.21E+07	3.06E+07	**5.04E+04**	**4.79E+06**	**2.99E+07**
Rank	−	−	−	+	+	+
f7						
Best	1.77E+04	1.80E+06	8.10E+06	**7.19E+03**	**8.86E+05**	**4.80E+06**
Worst	3.05E+04	2.16E+06	9.50E+06	9.46E+03	1.03E+06	5.37E+06
Mean	2.56E+04	1.99E+06	8.91E+06	**8.44E+03**	**9.70E+05**	**5.10E+06**
Std	3.51E+03	8.91E+04	3.94E+05	**6.02E+02**	**4.38E+04**	**1.54E+05**
Rank	−	−	−	+	+	+

Table 13 (continued)

F	Jaya algorithm			IJaya algorithm		
	n = 100	n = 500	n = 1000	n = 100	n = 500	n = 1000
f8						
Best	7.81E+04	8.17E+19	2.33E+22	**6.69E+02**	**2.38E+18**	**7.48E+21**
Worst	5.10E+13	2.22E+20	9.13E+22	9.80E+02	8.71E+18	1.32E+22
Mean	8.83E+12	1.36E+20	5.94E+22	**8.90E+02**	**6.31E+18**	**1.08E+22**
Std	1.26E+13	3.80E+19	1.84E+22	**6.88E+01**	**1.70E+18**	**1.66E+21**
Rank	−	−	−	+	+	+
f9						
Best	1.46E−05	4.26E−04	5.25E−04	**9.01E−07**	**2.98E−06**	**3.29E−05**
Worst	1.85E−03	6.06E−03	2.02E−02	2.52E−05	3.95E−04	7.39E−04
Mean	4.23E−04	2.84E−03	5.22E−03	**5.82E−06**	**9.78E−05**	**1.88E−04**
Std	5.15E−04	1.77E−03	4.48E−03	**6.11E−06**	**9.54E−05**	**1.52E−04**
Rank	−	−	−	+	+	+
f10						
Best	8.77E+11	7.14E+13	3.48E+14	**3.09E+11**	**3.95E+13**	**2.03E+14**
Worst	1.32E+12	8.95E+13	3.97E+14	4.41E+11	4.54E+13	2.39E+14
Mean	1.12E+12	8.28E+13	3.84E+14	**3.61E+11**	**4.24E+13**	**2.17E+14**
Std	1.25E+11	5.15E+12	1.36E+13	**2.98E+10**	**1.67E+12**	**7.99E+12**
Rank	−	−	−	+	+	+

5 Conclusion

In this paper, the Improved Jaya algorithm (IJaya) is proposed for global continuous optimization. The random walking phase of the original Jaya algorithm has been developed and thus the balance between exploration and exploitation is controlled. This modification has a significant impact on performance as it allows the algorithm to evade local optima by making occasional "jumps" in the search space of the algorithm while maintaining Jaya's original simple algorithm philosophy. Such behaviors are based on the use of randomly selected individuals during the random walking phase.

The performance of the proposed IJaya has been assessed on the eighteen unimodal and multimodal benchmark functions. The performance of IJaya has been tested not only on low scaled dimensions but also on large scale dimensions (Dimension = {10, 20, 30, 100, 500, and 1000}). In addition to different dimensions, the success of IJaya in different population sizes is shown in this paper. Real-world problems are mostly large-scale dimensional. Optimization algorithms change in their success as the dimension of the problem increases. That's why Jaya and IJaya are tested for performance not only in low dimensions but also in higher dimensions. Thus, the success of IJaya has

Table 14 The results of the Jaya and IJaya algorithms for the dimensions of 100, 500, and 1000 in multimodal Benchmark functions (Population size = 50)

F	Jaya algorithm			IJaya algorithm		
	n = 100	n = 500	n = 1000	n = 100	n = 500	n = 1000
f11						
Best	− 2.40E+01	**− 1.00E+02**	**− 1.91E+02**	**− 2.73E+01**	− 9.86E+01	− 1.85E+02
Worst	− 1.99E+01	− 8.80E+01	− 1.77E+02	− 2.38E+01	− 9.08E+01	− 1.75E+02
Mean	− 2.25E+01	**− 9.48E+01**	**− 1.83E+02**	**− 2.55E+01**	**− 9.48E+01**	− 1.80E+02
Std	1.14E+00	3.03E+00	3.81E+00	**8.95E−01**	**2.21E+00**	**2.88E+00**
Rank	−	=	+	+	=	−
f12						
Best	**− 1.84E+06**	**− 2.45E+08**	**1.27E+12**	3.13E+06	1.82E+11	6.68E+12
Worst	9.80E+07	2.37E+11	1.13E+13	8.11E+07	2.71E+11	1.31E+13
Mean	**2.66E+07**	**6.36E+10**	**7.13E+12**	3.50E+07	2.30E+11	8.52E+12
Std	3.02E+07	5.94E+10	1.76E+12	**1.96E+07**	**2.17E+10**	**1.35E+12**
Rank	+	+	+	−	−	−
f13						
Best	1.14E+02	1.61E+03	4.18E+03	**5.38E+01**	**7.71E+02**	**1.75E+03**
Worst	2.38E+02	2.17E+03	4.67E+03	6.92E+01	9.80E+02	2.56E+03
Mean	1.82E+02	1.95E+03	4.43E+03	**5.94E+01**	**8.56E+02**	**2.25E+03**
Std	3.55E+01	1.51E+02	1.58E+02	**4.21E+00**	**5.38E+01**	**1.45E+02**
Rank	−	−	−	+	+	+
f14						
Best	− 2.66E+03	− 9.27E+03	− 9.54E+03	**− 4.07E+04**	**− 6.68E+04**	**− 9.34E+04**
Worst	8.00E+02	8.66E+03	1.06E+04	− 1.34E+04	− 1.48E+04	− 2.46E+04
Mean	− 6.05E+02	− 2.21E+03	− 1.36E+03	**− 2.43E+04**	**− 3.56E+04**	**− 4.45E+04**
Std	1.13E+03	3.79E+03	5.43E+03	**1.10E+03**	**1.52E+03**	**1.90E+03**
Rank	−	−	−	+	+	+
f15						
Best	1.12E++03	6.86E+03	1.35E+04	**9.28E+02**	**5.89E+03**	**1.23E+04**
Worst	1.34E+03	7.45E+03	1.50E+04	1.04E+03	6.20E+03	1.29E+04
Mean	1.27E+03	7.12E+03	1.45E+04	**9.75E+02**	**6.05E+03**	**1.26E+04**
Std	4.86E+01	1.96E+02	3.77E+02	**2.94E+01**	**1.03E+02**	**1.69E+02**
Rank	−	−	−	+	+	+
f16						
Best	2.00E+01	2.00E+01	2.00E+01	**1.35E+01**	**1.85E+01**	**1.90E+01**
Worst	2.01E+01	2.07E+01	2.08E+01	1.48E+01	1.90E+01	1.95E+01
Mean	2.00E+01	2.01E+01	2.01E+01	**1.43E+01**	**1.88E+01**	**1.93E+01**
Std	4.66E−02	2.06E−01	2.38E−01	**2.92E−02**	**1.38E−01**	**1.04E−01**
Rank	−	−	−	+	+	+
f17						
Best	3.38E+02	6.93E+03	1.53E+04	**1.67E+02**	**3.66E+03**	**8.77E+03**
Worst	7.51E+02	8.69E+03	1.75E+04	2.42E+02	4.26E+03	1.06E+04
Mean	5.82E+02	7.52E+03	1.62E+04	**1.99E+02**	**3.97E+03**	**9.75E+03**
Std	1.05E+02	4.34E+02	5.96E+02	**1.90E+01**	**1.55E+02**	**4.82E++02**
Rank	−	−	−	+	+	+

Table 14 (continued)

F	Jaya algorithm			IJaya algorithm		
	n = 100	n = 500	n = 1000	n = 100	n = 500	n = 1000
f18						
Best	4.03E+02	9.09E+02	1.29E+03	**1.77E+02**	**5.01E+02**	**7.06E+02**
Worst	4.13E+02	9.17E+02	1.30E+03	2.60E+02	6.11E+02	8.51E+02
Mean	4.09E+02	9.14E+02	1.29E+03	**2.27E+02**	**5.45E+02**	**7.76E+02**
Std	2.55E+00	2.03E+00	2.18E+00	**1.86E+00**	**203E+00**	**2.18E+00**
Rank	−	−	−	+	+	+

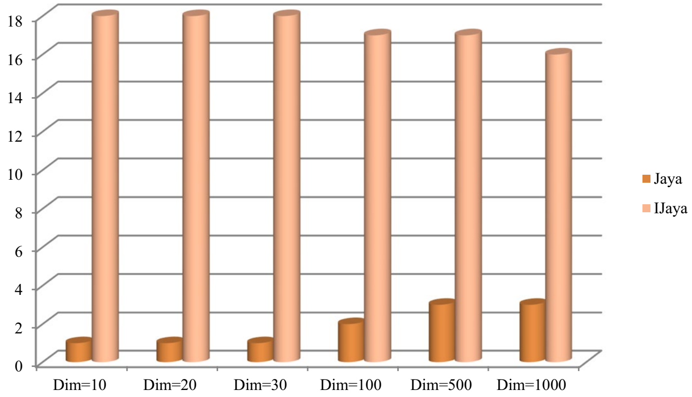
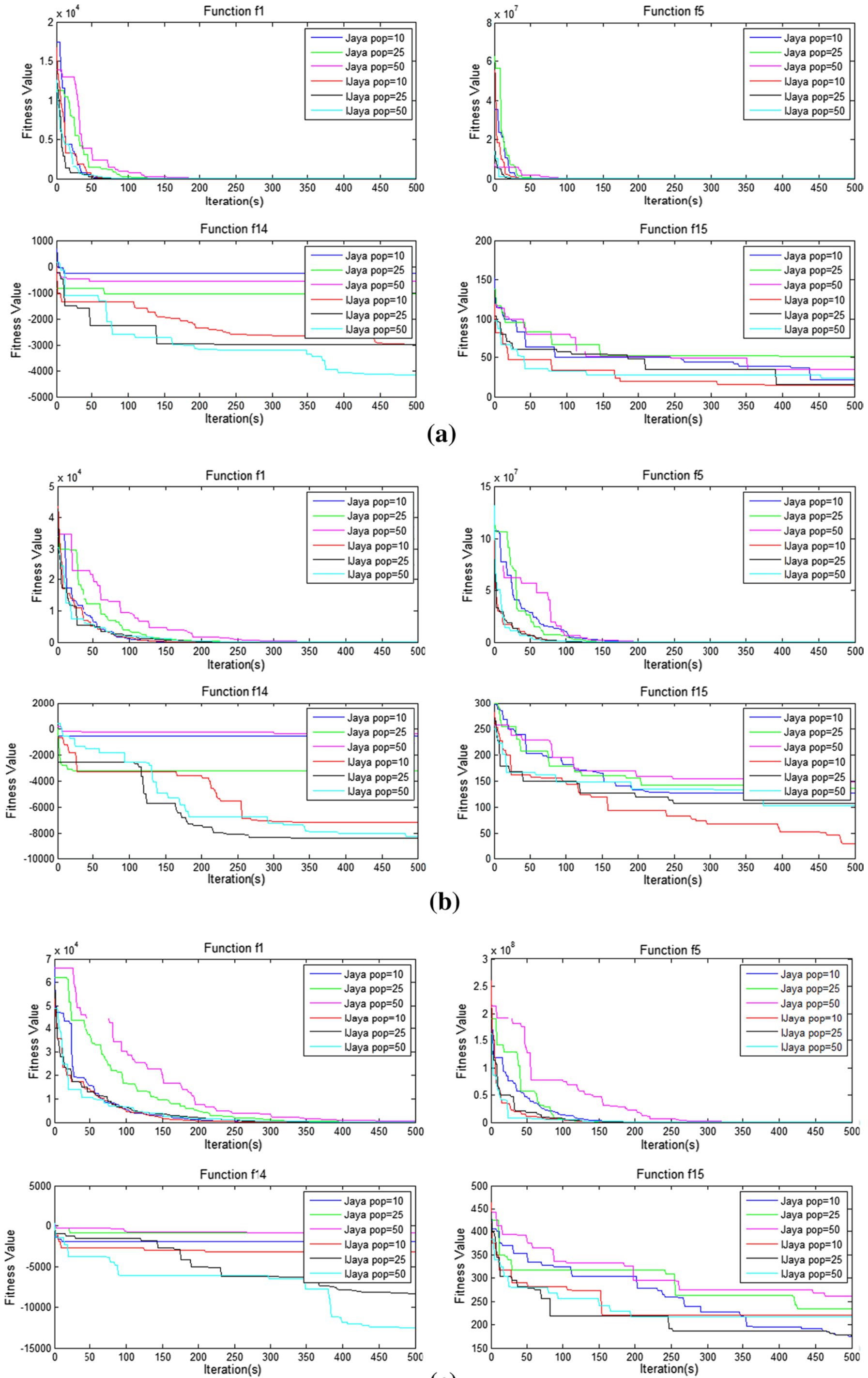


Fig. 4 Rank performances of Jaya and IJaya in terms of the mean results for the population size of 50 based on dimension sizes (Dim = dimension)

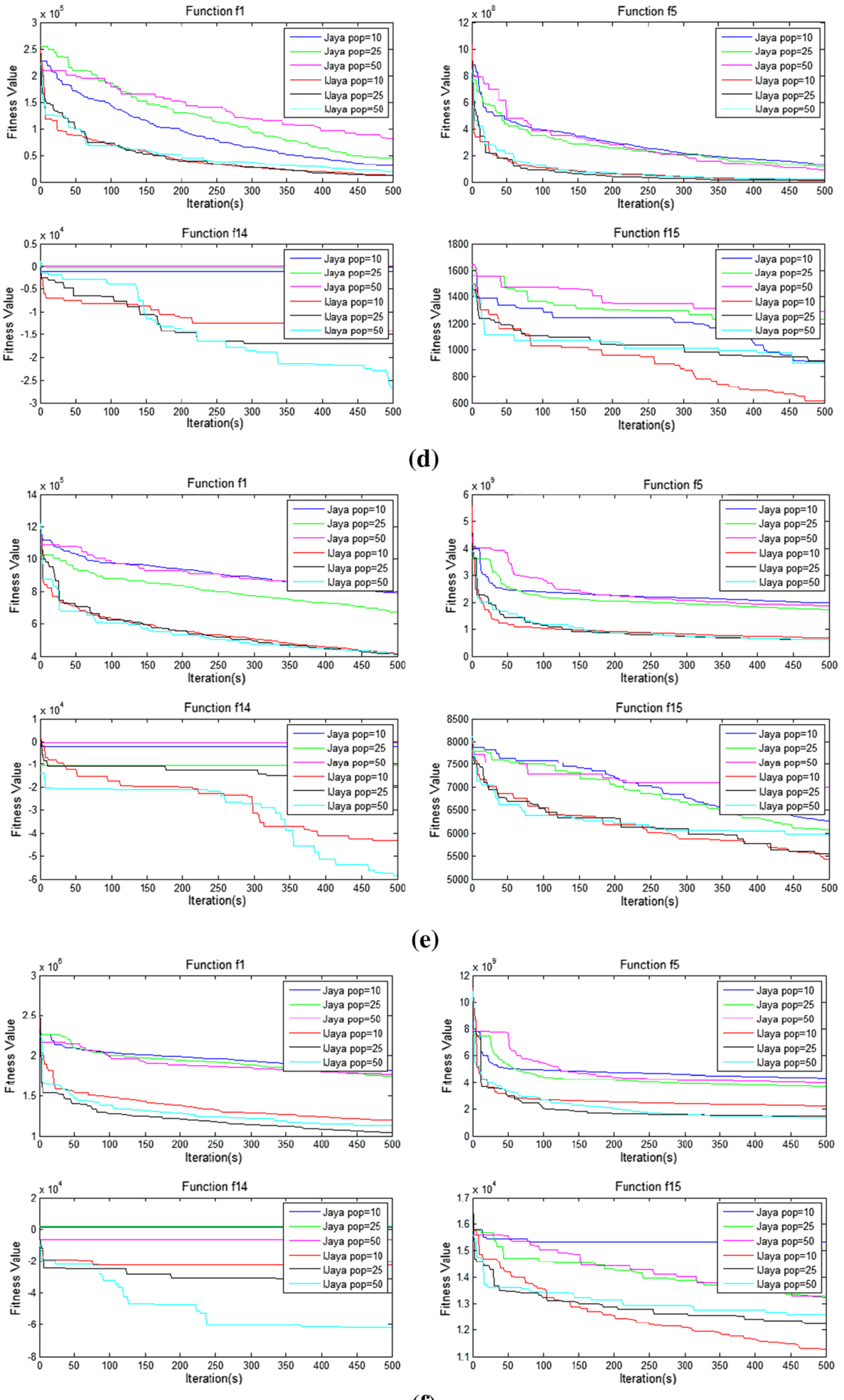
Fig. 5 Convergence graphs of the f1, f5, f14, and f15 functions for the Jaya and IJaya algorithms **a** for dimension = 10, **b** for dimension = 20, **c** for dimension = 30, **d** for dimension = 100, **e** for dimension = 500, and **f** for dimension = 1000 ▶


Function f1
Function f5
Function f14
Function f15
x 10^4
x 10^7
x 10^8
Fitness Value
Iteration(s)
0 50 100 150 200 250 300 350 400 450 500
Jaya pop=10
Jaya pop=25
Jaya pop=50
IJaya pop=10
IJaya pop=25
IJaya pop=50

(a)

(b)

(c)



(d)

(e)

(f)

Fig. 5 (continued)

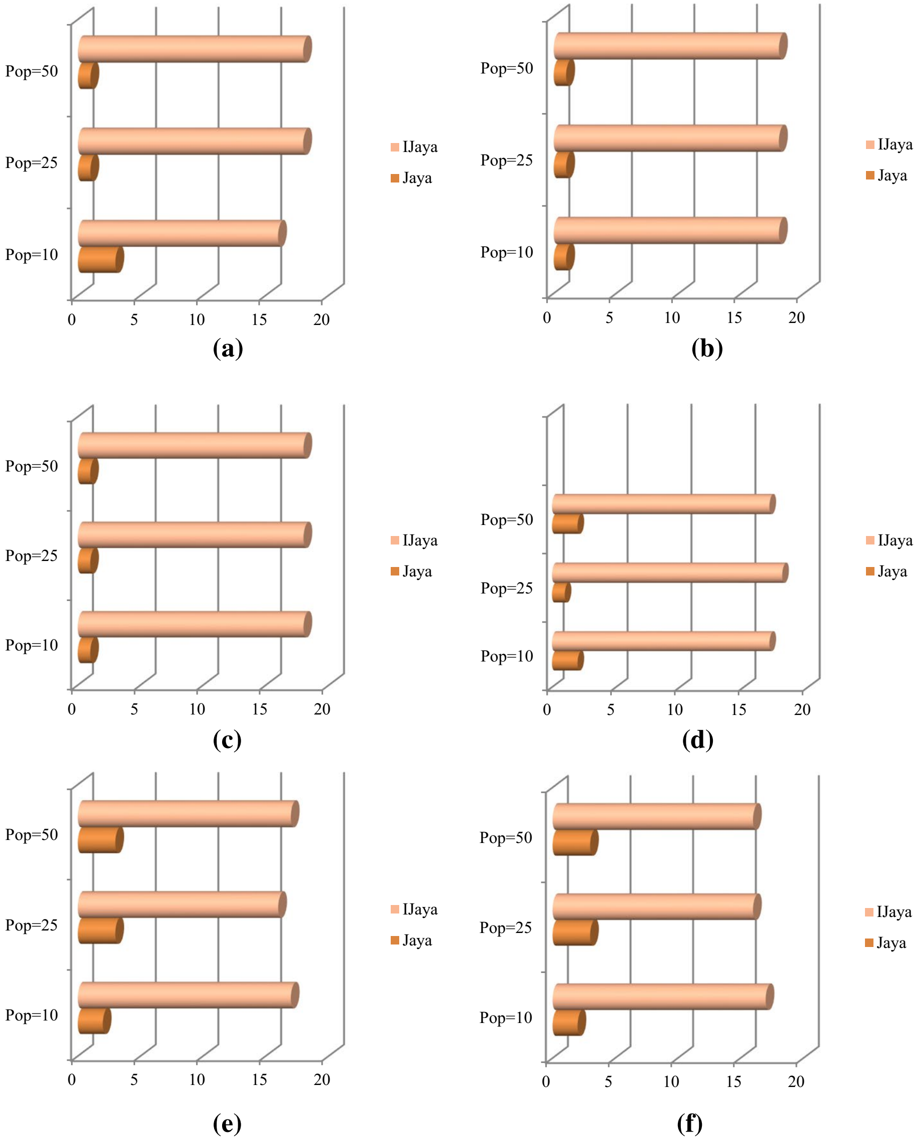


Fig. 6 Rank performances of Jaya and IJaya in terms of the mean results for the population size of 10, 25, and 50 **a** for dimension = 10, **b** for dimension = 20, **c** for dimension = 30, **d** for dimension = 100, **e** for dimension = 500, and **f** for dimension = 1000

Table 15 The statistical results of the Jaya and IJaya algorithms using Wilcoxon signed rank test for dimensions of 10 and 20

F	Dimension = 10 (IJaya- Jaya)						Dimension = 20 (IJaya- Jaya)					
	Pop = 10		Pop = 25		Pop = 50		Pop = 10		Pop = 25		Pop = 50	
	P-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign
f1	0.191334	(−)	8.86E−05	(+)	8.86E−05	(+)	0.0015073	(+)	0.002495	(+)	8.86E−05	(+)
f2	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f3	1	(−)	1	(−)	1	(−)	1	(−)	1	(−)	1	(−)
f4	0.1454	(−)	0.0099964	(+)	8.86E−05	(+)	0.3702613	(−)	8.86E−05	(+)	0.0003385	(+)
f5	0.008968	(+)	0.027621	(+)	0.0015073	(+)	0.0080344	(+)	0.1453998	(−)	0.0001033	(+)
f6	0.000103	(+)	8.86E−05	(+)	8.86E−05	(+)	0.0004493	(+)	0.0001033	(+)	0.0001204	(+)
f7	0.001019	(+)	8.86E−05	(+)	8.86E−05	(+)	0.0071892	(+)	8.86E−05	(+)	8.86E−05	(+)
f8	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f9	0.000103	(+)	0.0099964	(+)	8.86E−05	(+)	0.0123742	(+)	0.2043302	(−)	0.0003385	(+)
f10	0.000449	(+)	8.86E−05	(+)	8.86E−05	(+)	0.0003902	(+)	8.86E−05	(+)	8.86E−05	(+)
f11	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	0.0019443	(+)	8.86E−05	(+)	8.86E−05	(+)
f12	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.03E−03	(+)	8.86E−05	(+)
f13	0.000863	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	6.20E−02	(−)	8.86E−05	(+)
f14	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f15	0.004045	(+)	0.0010188	(+)	0.0001204	(+)	0.0089676	(+)	0.002495	(+)	8.86E−05	(+)
f16	0.000593	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f17	0.012374	(+)	0.1353569	(−)	0.0005934	(+)	0.3702613	(−)	0.001713	(+)	8.86E−05	(+)
f18	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)

Table 16 The statistical results of the Jaya and IJaya algorithms using Wilcoxon Signed Rank Test for dimensions of 30 and 100

F	Dimension = 30 (IJaya- Jaya)						Dimension = 100 (IJaya- Jaya)					
	Pop = 10		Pop = 25		Pop = 50		Pop = 10		Pop = 25		Pop = 50	
	P-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign
f1	0.0004493	(+)	0.0001033	(+)	8.86E−05	(+)	8.84E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f2	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f3	1	(−)	1	(−)	1	(−)	1	(−)	1	(−)	1	(−)
f4	0.0040452	(+)	8.86E−05	(+)	0.0303651	(+)	0.0001033	(+)	8.86E−05	(+)	8.86E−05	(+)
f5	8.86E−05	(+)	7.65E−01	(−)	6.20E−02	(−)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f6	0.0002536	(+)	0.0015073	(+)	0.005734	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f7	0.0001204	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f8	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f9	0.0001401	(+)	0.0438037	(+)	0.0001033	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f10	0.0011624	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f11	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	3.38E−04	(+)	8.86E−05	(+)
f12	8.86E−05	(+)	9.11E−01	(−)	4.49E−04	(+)	8.86E−05	(+)	5.50E−01	(−)	2.79E−01	(−)
f13	8.86E−05	(+)	1.00E+00	(−)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f14	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f15	0.0035918	(+)	0.0003385	(+)	8.86E−05	(+)	0.0003385	(+)	8.86E−05	(+)	8.86E−05	(+)
f16	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f17	0.0028209	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f18	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)

Table 17 The statistical results of the Jaya and IJaya algorithms using Wilcoxon Signed Rank Test for dimensions of 500 and 1000

F	Dimension = 500 (IJaya- Jaya)						Dimension = 1000 (IJaya- Jaya)					
	Pop = 10		Pop = 25		Pop = 50		Pop = 10		Pop = 25		Pop = 50	
	P-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign
f1	8.84E−05	(+)	8.84E−05	(+)	8.84E−05	(+)	8.83E−05	(+)	8.84E−05	(+)	8.81E−05	(+)
f2	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)
f3	1	(−)	1	(−)	1	(−)	1	(−)	1	(−)	1	(−)
f4	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)
f5	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)
f6	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)
f7	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)
f8	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)
f9	0.0001204	(−)	8.84E−05	(+)	8.84E−05	(+)	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)
f10	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)	0.8519246	(−)	8.84E−05	(+)	8.84E−05	(+)
f11	0.1560036	(−)	0.1168876	(−)	0.940481	(−)	0.0010188	(+)	0.4330483	(−)	0.005111	(+)
f12	0.1453998	(−)	0.1353569	(−)	8.84E−05	(+)	8.86E−05	(+)	0.011129	(+)	0.0123742	(+)
f13	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)
f14	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)
f15	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)
f16	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)
f17	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)
f18	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)

Table 18 Parameters of BA, PSO, Jaya, and IJaya algorithms

Algorithms	Parameters	Values
BA (Yang 2011)	*Qmin*	0
	Qmax	2
	alpha (Loudness)	0.5
	Gamma (emission rate)	0.5
PSO (Xinchao 2010)	w_{max}	0.9
	w_{min}	0.2
	c1	2
	c2	2

been demonstrated in problems of different dimensions. The success of IJaya has been also compared with the PSO, BA, CSS, GOA, MVO, and SSA algorithms in addition to the original Jaya algorithm in the low and large scale dimensions in the optimization problems. In addition, Jaya and IJaya have been applied to three famous mechanical engineering design problems which include the Welded Beam Design problem (WBD), the Compression Spring Design (CSD), and the Pressure Vessel Design problem (PVD). These problems consist of both equality and inequality constraints. Jaya and IJaya have been tested in detail in engineering design problems at three different population sizes (N = 20, 40, and 60) and three different amounts of MaxFEs (MaxFEs = 30,000, 50,000, and 100,000). Comparison results also showed that IJaya performed superiorly and can be used as an alternative algorithm for constrained and unconstrained continuous optimization. These superior results confirm the advantage of the updated random walking phase to escape local optima. Apart from the specific numerical results reported here, one of the main advantages of IJaya is its simplicity. IJaya is much simpler to implement (IJaya has two parameters, i.e. the population size and the number of generations) than most of the compared algorithms from the state-of-the-art.

In future research, it is thought to test the success of the Jaya algorithm not only in continuous optimization but also on discrete and binary optimization problems by hybridizing different algorithms.

Table 19 The results of the IJaya and other algorithms for the dimensions of 10 and 20

	BA		PSO		Jaya		IJaya	
	n = 10	n = 20	n = 10	n = 20	n = 10	n = 20	n = 10	n = 20
f1								
Best	3.50E+03	9.01E+03	1.02E+04	3.25E+04	5.95E−07	5.22E−02	**5.20E−08**	**4.72E−02**
Worst	1.28E+05	5.58E+05	1.38E+05	5.33E+05	1.39E−05	3.64E+00	1.13E−06	3.48E−01
Mean	1.24E+04	3.62E+04	1.67E+04	4.22E+04	3.83E−06	5.67E−01	**2.51E−07**	**1.22E−01**
Std	3.61E+03	8.00E+03	3.31E+03	4.64E+03	2.83E−06	8.45E−01	**2.34E−07**	**7.45E−02**
Rank	3	3	4	4	2	2	**1**	**1**
f2								
Best	1.15E+04	2.37E+05	3.47E+04	3.06E+05	5.65E+04	3.96E+05	**8.41E−07**	**4.41E−03**
Worst	1.28E+05	5.58E+05	1.38E+05	5.33E+05	2.40E+05	9.92E+05	6.93E−02	9.89E+00
Mean	8.67E+04	4v.25E+05	7.07E+04	4.08E+05	1.56E+05	6.17E+05	**9.15E−03**	**1.07E+00**
Std	2.92E+04	9.81E+04	2.44E+04	6.72E+04	4.60E+04	1.67E+05	**1.90E−02**	**2.19E+00**
Rank	3	3	2	2	4	4	**1**	**1**
f3								
Best	**0.00E+00**	**0.00E+00**	3.21E+00	3.15E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Worst	0.00E+00	0.00E+00	3.02E+03	1.56E+04	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Mean	**0.00E+00**	**0.00E+00**	7.33E+00	4.85E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Std	**0.00E+00**	**0.00E+00**	2.75E+00	1.09E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Rank	**1**	**1**	2	2	**1**	**1**	**1**	**1**
f4								
Best	6.60E+01	5.19E+02	5.11E+02	5.36E+03	5.37E−04	1.34E−02	**2.10E−04**	**1.52E−02**
Worst	1.05E+03	4.62E+03	7.33E+07	2.08E+08	1.04E−02	8.54E−01	3.30E−03	4.55E−01
Mean	3.90E+02	2.12E+03	1.60E+03	9.25E+03	2.98E−03	1.77E−01	**1.20E−03**	**1.26E−01**
Std	2.52E+02	9.46E+02	7.46E+02	2.53E+03	2.70E−03	1.98E−01	**8.08E−04**	**1.11E−01**
Rank	3	3	4	4	2	2	**1**	**1**

Table 19 (continued)

	BA		PSO		Jaya		IJaya	
	n = 10	n = 20	n = 10	n = 20	n = 10	n = 20	n = 10	n = 20
f5								
Best	1.31E+06	3.13E+06	4.54E+06	9.34E+07	1.55E−02	1.52E+01	**1.50E−02**	**1.09E+01**
Worst	3.67E+07	1.48E+08	7.33E+07	2.08E+08	1.49E+02	2.23E+02	3.01E+01	1.72E+02
Mean	1.76E+07	7.33E+07	4.30E+07	1.43E+08	1.75E+01	9.39E+01	**4.98E+00**	**6.60E+01**
Std	1.01E+07	3.67E+07	1.97E+07	3.73E+07	3.25E+01	5.41E+01	**7.13E+00**	**4.51E+01**
Rank	3	3	4	4	2	2	**1**	**1**
f6								
Best	2.23E+03	2.88E+03	3.26E+04	3.20E+05	6.21E−01	1.28E+00	**1.92E−02**	**1.88E−01**
Worst	1.21E+05	5.04E+05	1.82E+05	1.15E+06	9.59E+01	8.28E+01	4.16E−01	7.44E+00
Mean	4.33E+04	2.62E+05	8.93E+04	7.20E+05	2.17E+01	5.13E+01	**1.60E−01**	**3.26E+00**
Std	3.30E+04	1.45E+05	3.88E+04	2.00E+05	2.68E+01	2.29E+01	**9.41E−02**	**2.15E+00**
Rank	3	3	4	4	2	2	**1**	**1**
f7								
Best	8.96E+01	8.50E+02	3.71E+02	2.94E+03	4.98E−08	1.05E−02	**3.10E−09**	**4.34E−03**
Worst	8.02E+02	3.55E+03	1.05E+04	6.77E+06	1.58E−06	1.63E−01	4.81E−08	1.71E−02
Mean	4.17E+02	2.39E+03	8.60E+02	4.12E+03	3.77E−07	4.91E−02	**1.31E−08**	**9.05E−03**
Std	1.83E+02	8.77E+02	2.37E+02	6.87E+02	3.93E−07	3.63E−02	**1.27E−08**	**3.36E−03**
Rank	3	3	4	4	2	2	**1**	**1**
f8								
Best	3.61E+01	1.14E+02	6.84E+01	4.98E+02	3.13E+01	2.92E+02	**6.81E−02**	**3.17E+01**
Worst	6.04E+07	1.68E+10	1.05E+04	6.77E+06	2.41E+02	6.77E+02	9.57E−01	8.53E+01
Mean	1.16E+07	4.79E+09	9.09E+02	5.41E+05	1.04E+02	4.58E+02	**4.01E−01**	**5.76E+01**
Std	1.74E+07	4.51E+09	2.21E+03	1.50E+06	4.89E+01	1.29E+02	**1.92E−01**	**1.43E+01**
Rank	4	4	3	3	2	2	**1**	**1**

Table 19 (continued)

	BA		PSO		Jaya		IJaya	
	n = 10	n = 20	n = 10	n = 20	n = 10	n = 20	n = 10	n = 20
f9								
Best	0.00E+00	0.00E+00	8.46E−02	2.22E−01	7.19E−24	1.20E−16	**4.40E−24**	**1.27E−16**
Worst	1.68E−01	1.16E−01	4.52E+10	2.20E+11	4.21E−20	1.29E−11	3.93E−21	9.51E−13
Mean	3.43E−02	4.65E−02	3.98E−01	7.18E−01	5.06E−21	7.42E−13	**6.82E−22**	**1.19E−13**
Std	3.80E−02	2.95E−02	2.13E−01	2.67E−01	9.47E−21	2.80E−12	**9.69E−22**	**2.06E−13**
Rank	3	3	4	4	2	2	**1**	**1**
f10								
Best	1.99E+10	1.17E+11	2.04E+10	1.33E+11	2.06E+00	4.62E+05	**3.65E−02**	**1.27E+05**
Worst	5.06E+10	2.19E+11	4.52E+10	2.20E+11	3.91E+01	9.67E+06	1.80E+00	9.03E+05
Mean	3.44E+10	1.67E+11	3.39E+10	1.74E+11	1.45E+01	2.40E+06	**3.91E−01**	**4.51E+05**
Std	6.79E+09	2.38E+10	6.93E+09	2.65E+10	9.26E+00	2.13E+06	**4.04E−01**	**2.10E+05**
Rank	4	3	3	4	2	2	**1**	**1**
f11								
Best	− 4.77E+00	− 7.32E+00	− 3.11E+00	− 5.11E+00	− 6.61E+00	− 8.22E+00	− **8.74E+00**	− **1.06E+01**
Worst	3.89E+03	1.85E+05	1.39E+04	6.28E+05	− 3.58E+00	− 5.18E+00	− 6.70E+00	− 8.51E+00
Mean	− 3.21E+00	− 4.99E+00	− 2.60E+00	− 4.37E+00	− 4.95E+00	− 6.73E+00	− **7.78E+00**	− **9.41E+00**
Std	6.26E−01	8.14E−01	**2.84E−01**	**4.10E−01**	8.34E−01	8.21E−01	5.24E−01	5.20E−01
Rank	3	3	4	4	2	2	**1**	**1**
f12								
Best	− 3.59E+03	4.95E+04	4.91E+03	1.44E+05	− 4.32E+03	− 6.61E+04	− **5.14E+03**	− **6.73E+04**
Worst	3.89E+03	1.85E+05	1.39E+04	6.28E+05	3.83E+03	8.04E+04	− 3.64E+03	− 5.52E+03
Mean	1.18E+03	1.12E+05	8.19E+03	3.32E+05	− 1.38E+03	7.86E+03	− **4.54E+03**	− **2.18E+04**
Std	1.92E+03	3.56E+04	1.93E+03	1.88E+05	2.47E+03	4.23E+04	**3.45E+02**	**9.98E+03**
Rank	3	3	4	4	2	2	**1**	**1**

Table 19 (continued)

	BA		PSO		Jaya		IJaya	
	n = 10	n = 20	n = 10	n = 20	n = 10	n = 20	n = 10	n = 20
f13								
Best	8.34E+00	1.68E+01	1.65E+01	8.41E+01	1.49E−06	1.25E−01	**8.71E−09**	**6.62E−01**
Worst	3.51E+01	1.05E+02	6.74E+01	1.64E+02	4.54E−01	1.26E+01	1.22E−06	2.31E+00
Mean	1.84E+01	5.92E+01	4.18E+01	1.25E+02	2.27E−02	2.94E+00	**1.49E−07**	**1.16E+00**
Std	7.27E+00	1.89E+01	1.40E+01	2.20E+01	9.90E−02	3.21E+00	**2.63E−07**	**4.25E−01**
Rank	3	3	4	4	2	2	**1**	**1**
f14								
Best	− 2.07E+03	− 3.51E+03	− 1.84E+03	− 3.61E+03	− 1.17E+03	− 4.56E+03	**− 4.19E+03**	**− 8.38E+03**
Worst	1.27E+02	2.84E+02	1.39E+02	3.35E+02	1.76E+02	− 2.94E+02	− 1.91E+03	− 2.28E+03
Mean	− 1.16E+03	− 1.83E+03	− 1.79E+03	− 3.30E+03	− 6.90E+02	− 1.16E+03	**− 3.33E+03**	**− 6.80E+03**
Std	3.66E+02	7.74E+02	7.94E+02	2.07E+03	3.52E+02	8.61E+02	**3.51E+02**	**8.51E+02**
Rank	3	3	2	2	4	4	**1**	**1**
f15								
Best	3.82E+01	1.84E+02	9.61E+01	2.51E+02	2.53E+01	1.01E+02	**2.15E+01**	**9.49E+01**
Worst	1.27E+02	2.84E+02	1.39E+02	3.35E+02	5.83E+01	1.87E+02	4.02E+01	1.43E+02
Mean	9.90E+01	2.43E+02	1.25E+02	2.85E+02	3.99E+01	1.46E+02	**3.01E+01**	**1.21E+02**
Std	2.16E+01	3.10E+01	1.18E+01	1.85E+01	8.84E+00	2.52E+01	**5.09E+00**	**1.21E+01**
Rank	3	3	4	4	2	2	**1**	**1**
f16								
Best	1.63E+01	1.94E+01	1.85E+01	1.99E+01	4.61E−03	1.77E+01	**1.18E−04**	**1.02E+00**
Worst	1.68E+02	4.43E+02	2.00E+01	2.00E+01	1.78E+01	2.00E+01	8.50E−04	2.90E+00
Mean	1.94E+01	2.01E+01	1.98E+01	2.00E+01	9.19E+00	1.96E+01	**3.56E−04**	**2.08E+00**
Std	8.46E−01	3.14E−01	3.53E−01	**0.00E+00**	7.78E+00	6.85E−01	**1.74E−04**	4.55E−01
Rank	3	4	4	3	2	2	**1**	**1**

Table 19 (continued)

	BA		PSO		Jaya		IJaya	
	n = 10	n = 20	n = 10	n = 20	n = 10	n = 20	n = 10	n = 20
f17								
Best	8.19E+01	1.31E+02	8.92E+01	2.43E+02	2.70E−01	3.82E−01	**2.00E−01**	**2.30E−01**
Worst	1.68E+02	4.43E+02	1.87E+02	4.55E+02	9.03E−01	9.52E−01	6.50E−01	8.56E−01
Mean	1.33E+02	3.44E+02	1.49E+02	3.78E+02	6.06E−01	7.48E−01	**5.08E−01**	**5.57E−01**
Std	2.28E+01	6.98E+01	2.75E+01	4.43E+01	1.82E−01	1.53E−01	**1.12E−01**	**1.52E−01**
Rank	3	3	4	4	2	2	**1**	**1**
f18								
Best	9.00E+01	6.99E+01	**9.00E+00**	**6.91E+00**	8.47E+01	1.27E+02	3.32E+01	6.17E+01
Worst	1.70E+01	2.58E+01	1.57E+01	2.42E+01	9.53E+01	1.36E+02	4.84E+01	8.15E+01
Mean	1.48E+01	2.28E+01	**1.34E+01**	**2.18E+01**	9.12E+01	1.31E+02	4.01E+01	6.76E+01
Std	1.74E+00	1.50E+00	**1.59E+00**	**1.45E+00**	2.80E+00	2.62E+00	2.17E+00	2.37E+00
Rank	2	2	**1**	**1**	4	4	3	3

Table 20 The results of the IJaya and other algorithms for the dimensions of 30 and 100

	BA		PSO		Jaya		IJaya	
	n = 30	n = 100	n = 30	n = 100	n = 30	n = 100	n = 30	n = 100
f1								
Best	2.02E+04	6.93E+04	5.72E+04	2.51E+05	1.28E+01	2.65E+04	**5.02E+00**	**1.01E+04**
Worst	1.25E+06	1.54E+07	1.27E+06	1.53E+07	1.28E+02	4.88E+04	2.27E+01	1.64E+04
Mean	6.10E+04	2.36E+05	6.85E+04	2.77E+05	5.45E+01	3.86E+04	**1.35E+01**	**1.32E+04**
Std	1.11E+04	4.06E+04	5.80E+03	1.09E+04	2.84E+01	5.68E+03	**4.33E+00**	**1.82E+03**
Rank	3	3	4	4	2	2	**1**	**1**
f2								
Best	2.49E+05	6.35E+06	5.99E+05	1.11E+07	9.65E+05	1.23E+07	**1.52E+02**	**9.16E+04**
Worst	1.25E+06	1.54E+07	1.27E+06	1.53E+07	1.98E+06	1.84E+07	1.59E+04	1.02E+06
Mean	1.04E+06	1.32E+07	9.99E+05	1.36E+07	1.38E+06	1.59E+07	**2.77E+03**	**4.03E+05**
Std	2.05E+05	1.95E+06	1.59E+05	9.72E+05	3.31E+05	1.56E+06	**3.56E+03**	**2.86E+05**
Rank	3	2	2	3	4	4	**1**	**1**
f3								
Best	**0.00E+00**	**0.00E+00**	7.02E+01	1.60E+03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Worst	0.00E+00	0.00E+00	2.64E+04	1.40E+05	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Mean	**0.00E+00**	**0.00E+00**	1.19E+02	1.90E+03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Std	**0.00E+00**	**0.00E+00**	2.36E+01	1.48E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Rank	**1**	**1**	2	2	**1**	**1**	**1**	**1**
f4								
Best	9.97E+02	3.78E+03	8.67E+03	7.23E+04	8.69E−01	2.89E+03	**1.86E−01**	**1.33E+03**
Worst	6.21E+03	2.65E+04	3.41E+08	1.41E+09	6.64E+01	5.53E+03	1.44E+01	2.54E+03
Mean	3.38E+03	1.54E+04	1.64E+04	1.07E+05	7.94E+00	4.40E+03	**5.22E+00**	**1.96E+03**
Std	1.48E+03	5.71E+03	4.75E+03	1.74E+04	1.41E+01	6.97E+02	**3.27E+00**	**3.01E+02**
Rank	3	3	4	4	2	2	**1**	**1**
f5								
Best	1.32E+07	7.23E+07	1.48E+08	8.89E+08	3.03E+02	5.87E+07	**2.28E+02**	**7.94E+06**
Worst	2.15E+08	8.88E+08	3.41E+08	1.41E+09	4.42E+04	1.56E+08	5.90E+03	2.26E+07

Table 20 (continued)

	BA		PSO		Jaya		IJaya	
	n = 30	n = 100	n = 30	n = 100	n = 30	n = 100	n = 30	n = 100
Mean	1.33E+08	6.18E+08	2.66E+08	1.20E+09	4.76E+03	9.86E+07	**2.28E+03**	**1.36E+07**
Std	5.51E+07	2.18E+08	4.38E+07	1.19E+08	9.82E+03	2.23E+07	**1.49E+03**	**3.34E+06**
Rank	3	3	4	4	2	2	**1**	**1**
f6								
Best	9.05E+04	6.12E+05	1.42E+06	2.41E++07	1.43E+01	1.00E+06	**5.01E+00**	**2.14E+05**
Worst	1.41E+06	1.67E+07	2.58E+06	3.33E+07	2.22E+02	2.58E+06	1.01E+02	4.35E+05
Mean	7.46E+05	1.03E+07	1.94E+06	2.84E+07	7.84E+01	1.80E+06	**3.24E+01**	**2.97E+05**
Std	3.83E+05	4.88E+06	3.54E+05	2.20E+06	5.13E+01	4.12E+05	**2.14E+01**	**7.04E+04**
Rank	3	3	4	4	2	2	**1**	**1**
f7								
Best	3.26E+03	3.19E+04	7.36E+03	1.27E+05	2.03E+00	7.52E+03	**6.15E−01**	**4.05E+03**
Worst	8.44E+03	1.11E+05	1.24E+09	5.16E+13	2.19E+01	2.63E+04	4.19E+00	6.01E+03
Mean	6.33E+03	7.89E+04	9.69E+03	1.39E+05	7.49E+00	1.57E+04	**1.64E+00**	**4.90E+03**
Std	1.77E+03	2.61E+04	1.12E+03	7.36E+03	4.65E+00	4.20E+03	**8.07E−01**	**5.84E+02**
Rank	3	3	4	4	2	2	**1**	**1**
f8								
Best	1.65E+02	7.32E+02	1.43E+03	6.13E+10	3.83E+02	3.52E+12	**1.05E+02**	**7.86E+02**
Worst	5.56E+11	1.13E+16	1.24E+09	5.16E+13	4.75E+08	2.35E+14	2.01E+02	1.10E+03
Mean	2.12E+11	4.55E+15	1.14E+08	1.25E+13	2.38E+07	4.74E+13	**1.57E+02**	**9.18E+02**
Std	1.63E+11	3.84E+15	3.03E+08	1.46E+13	1.04E+08	5.47E+13	**2.55E+01**	**5.06E+00**
Rank	4	4	3	2	2	3	**1**	**1**
f9								
Best	0.00E+00	5.79E−03	4.49E−01	8.82E−01	3.67E−13	7.63E−06	**3.54E−13**	**3.75E−08**
Worst	1.35E−01	2.15E−01	5.57E+11	6.44E+12	1.16E−07	1.21E−03	1.78E−09	4.14E−06
Mean	4.49E−02	6.18E−02	9.11E−01	1.44E+00	1.12E−08	2.80E−04	**2.28E−10**	**1.12E−06**
Std	4.15E−02	5.01E−02	1.99E−01	3.27E−01	2.74E−08	3.43E−04	**3.87E−10**	**1.161E−06**

Table 20 (continued)

	BA		PSO		Jaya		IJaya	
	n = 30	n = 100	n = 30	n = 100	n = 30	n = 100	n = 30	n = 100
Rank	3	3	4	4	2	2	**1**	**1**
f10								
Best	2.93E+11	4.51E+12	2.79E+11	4.96E+12	7.82E+07	3.72E+11	**3.07E+07**	**1.52E+11**
Worst	5.21E+11	6.04E+12	5.57E+11	6.44E+12	8.93E+08	1.04E+12	2.17E+08	2.75E+11
Mean	4.33E+11	5.37E+12	4.42E+11	5.75E+12	3.10E+08	6.47E+11	**6.79E+07**	**2.03E+11**
Std	5.50E+10	4.29E+11	6.40E+10	3.67E+11	1.95E+08	1.61E+11	**3.84E+07**	**2.73E+10**
Rank	3	3	4	4	2	2	**1**	**1**
f11								
Best	− 8.62E+00	− 2.07E+01	− 7.54E+00	− 1.93E+01	− 1.03E+01	− 2.72E+01	**− 1.39E+01**	**− 2.80E+01**
Worst	1.33E+06	6.06E+08	5.62E+06	2.74E+09	− 6.86E+00	− 1.98E+01	− 1.07E+01	− 2.34E+01
Mean	− 6.86E+00	− 1.79E+01	− 5.92E+00	− 1.55E+01	− 8.87E+00	− 2.28E+01	**− 1.16E+01**	**− 2.54E+01**
Std	9.37E−01	1.32E+00	7.50E−01	1.25E+00	8.90E−01	1.72E+00	**7.46E−01**	**1.13E+00**
Rank	3	3	4	4	2	2	**1**	**1**
f12								
Best	1.00E+05	1.73E+08	7.56E+05	2.05E+09	− 2.51E+05	**− 2.06E+07**	**− 2.34E+05**	6.24E+05
Worst	1.33E+06	6.06E+08	5.62E+06	2.74E+09	4.94E+05	8.30E+07	4.24E+05	9.65E+07
Mean	8.11E+05	3.55E+08	4.60E+06	2.39E+09	1.43E+05	3.51E+07	**1.59E+05**	**3.10E+07**
Std	2.68E+05	1.14E+08	9.94E+05	1.89E+08	2.68E+05	3.31E+07	**1.14E+05**	**2.20E+07**
Rank	3	3	4	4	2	2	**1**	**1**
f13								
Best	9.00E+00	2.57E+02	1.19E+02	9.12E+02	1.70E+00	8.38E+01	**1.59E+00**	**3.62E+01**
Worst	1.33E+02	5.86E+02	2.79E+02	1.09E+03	2.83E+01	1.94E+02	1.62E+01	5.28E+01
Mean	9.21E+01	4.10E+02	2.15E+02	9.99E+02	1.23E+01	1.34E+02	**1.17E+01**	**4.60E+01**
Std	3.10E+01	7.82E+01	4.01E+01	5.29E+01	7.37E+00	2.45E+01	**2.11E+00**	**5.02E+00**
Rank	3	3	4	4	2	2	**1**	**1**

Table 20 (continued)

	BA		PSO		Jaya		IJaya	
	n = 30	n = 100	n = 30	n = 100	n = 30	n = 100	n = 30	n = 100
f14								
Best	− 3.79E+03	− 8.84E+03	− 5.06E+03	− 1.48E+04	− 3.55E+03	− 2.88E+03	**− 1.20E+04**	**− 2.84E+04**
Worst	4.58E+02	1.63E+03	4.92E+02	1.71E+03	1.66E+02	1.29E+03	− 2.97E+03	− 5.23E+03
Mean	− 2.37E+03	− 6.04E+03	− 4.59E+03	− 1.36E+04	− 1.06E+03	− 1.50E+03	**− 8.79E+03**	**− 1.36E+04**
Std	6.08E+02	1.48E+03	3.45E+02	5.53E+03	8.06E+02	1.06E+03	**8.01E+02**	**1.03E+03**
Rank	3	2	2	4	4	3	**1**	**1**
f15								
Best	3.17E+02	1.28E+03	3.71E+02	1.53E+03	1.97E+02	9.69E+02	**1.82E+02**	**8.29E+02**
Worst	4.58E+02	1.63E+03	4.92E+02	1.71E+03	3.17E+02	1.42E+03	2.49E+02	1.02E+03
Mean	4.00E+02	1.50E+03	4.47E++02	1.63E+03	2.58E++02	1.21E+03	**2.24E+02**	**9.36E+02**
Std	3.13E+01	8.41E+01	2.86E+01	4.67E+01	3.12E+01	1.04E+02	**1.68E+01**	**5.11E+01**
Rank	3	3	4	4	2	2	**1**	**1**
f16								
Best	1.97E+01	1.97E+01	2.00E+01	2.00E+01	1.92E+01	2.00E+01	**5.55E+00**	**1.16E+01**
Worst	6.49E+02	2.44E+03	2.00E+01	2.00E+01	2.00E+01	2.04E+01	8.87E+00	1.37E+01
Mean	2.03E+01	2.02E+01	2.00E+01	2.00E+01	1.99E+01	2.00E+01	**7.05E+00**	**1.28E+01**
Std	2.46E−01	2.53E−01	**0**	**0**	1.72E−01	9.24E−02	1.18E−01	5.81E−02
Rank	4	3	3	2	2	2	**1**	**1**
f17								
Best	4.77E+02	1.86E+03	5.02E+02	2.18E+03	1.14E+00	1.76E+02	**1.07E+00**	**9.21E+01**
Worst	6.49E+02	2.44E+03	7.08E+02	2.72E+03	2.84E+00	5.65E+02	1.22E+00	1.67E+02
Mean	5.71E+02	2.25E+03	6.15E+02	2.47E+03	1.64E+00	3.76E+02	**1.13E+00**	**1.20E+02**
Std	4.88E+01	1.42E+02	5.49E+01	1.15E+02	4.97E−01	9.17E+01	**3.72E−02**	**1.84E+01**
Rank	3	3	4	4	2	2	**1**	**1**
f18								
Best	2.54E+01	**4.82E+01**	**2.39E+01**	4.94E+01	1.53E+02	2.87E+02	7.36E+01	1.48E+02

Table 20 (continued)

	BA		PSO		Jaya		IJaya	
	n = 30	n = 100	n = 30	n = 100	n = 30	n = 100	n = 30	n = 100
Worst	3.17E+01	5.71E+01	2.88E+01	5.60E+01	1.65E+02	2.95E+02	1.00E+02	1.76E+02
Mean	2.78E+01	**5.22E+01**	**2.69E+01**	5.29E+01	1.59E+02	2.91E+02	8.57E+01	1.60E+02
Std	1.60E+00	1.97E+00	**1.40E+00**	**1.66E+00**	2.96E+00	2.03E+00	2.11E+00	2.02E+00
Rank	2	**1**	**1**	2	4	4	3	3

Table 21 The results of the IJaya and other algorithms for the dimensions of 500 and 1000

	BA		PSO		Jaya		IJaya	
	n = 500	n = 1000	n = 500	n = 1000	n = 500	n = 1000	n = 500	n = 1000
f1								
Best	3.93E+05	9.90E+05	1.46E+06	3.08E+06	7.08E+05	1.62E+06	**3.50E+05**	**9.84E+05**
Worst	4.24E+08	1.48E+09	3.96E+08	1.60E+09	8.56E+05	1.82E+06	4.61E+05	1.17E+06
Mean	1.29E+06	2.55E+06	1.54E+06	3.16E+06	7.73E+05	1.73E+06	**4.09E+05**	**1.06E+06**
Std	2.23E+05	5.50E+05	3.01E+04	5.19E+04	3.37E+04	5.21E+04	**2.73E+04**	**5.00E+04**
Rank	3	3	4	4	2	2	**1**	**1**
f2								
Best	8.47E+07	4.13E+08	3.63E+08	1.48E+09	3.82E+08	1.59E+09	**3.84E+07**	**3.38E+08**
Worst	4.24E+08	1.48E+09	3.96E+08	1.60E+09	4.47E+08	1.75E+09	1.10E+08	5.48E+08
Mean	3.37E+08	1.34E+09	3.83E+08	1.56E+09	4.11E+08	1.67E+09	**7.36E+07**	**4.36E+08**
Std	6.36E+07	2.18E+08	9.14E+07	4.97E+07	1.56E+07	4.93E+07	**1.56E+07**	**4.91E+07**
Rank	2	2	3	3	4	4	**1**	**1**
f3								
Best	**0.00E+00**	**0.00E+00**	5.37E+04	2.31E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Worst	0.00E+00	0.00E+00	8.55E+05	1.76E+06	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Mean	**0.00E+00**	**0.00E+00**	5.90E+04	2.45E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Std	**0.00E+00**	**0.00E+00**	1.93E+03	6.20E+03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Rank	**1**	**1**	2	2	**1**	**1**	**1**	**1**
f4								
Best	4.22E+04	8.51E+04	7.05E+05	1.54E+06	6.00E+04	1.39E+05	**3.54E+04**	**8.25E+04**
Worst	1.59E+05	3.00E+05	7.52E+09	1.54E+10	7.11E+04	1.51E+05	4.77E+04	1.13E+05
Mean	9.45E+04	1.95E+05	7.68E+05	1.64E+06	6.50E+04	1.46E+05	**4.11E+04**	**9.90E+04**
Std	3.16E+04	6.05E+04	4.10E+04	6.91E+04	3.32E+03	3.51E+03	**3.24E+03**	**3.51E+03**
Rank	3	4	4	3	2	2	**1**	**1**

Table 21 (continued)

	BA		PSO		Jaya		IJaya	
	n = 500	n = 1000	n = 500	n = 1000	n = 500	n = 1000	n = 500	n = 1000
f5								
Best	5.57E+08	9.26E+09	6.71E+09	1.35E+10	1.48E+09	3.75E+09	**5.45E+08**	**1.40E+09**
Worst	5.87E+09	9.11E+09	7.52E+09	1.54E+10	2.01E+09	4.34E+09	8.52E+08	2.08E+09
Mean	2.02E+09	5.24E+09	7.20E+09	1.47E+10	1.73E+09	4.03E+09	**6.56E+08**	**1.68E+09**
Std	2.03E+09	2.55E+09	2.06E+08	4.70E+08	1.37E+08	1.43E+08	**8.58E+07**	**1.01E+08**
Rank	3	3	4	4	2	2	**1**	**1**
f6								
Best	5.09E+07	8.85E+08	8.43E+08	3.46E+09	1.84E+08	8.24E+08	**5.07E+07**	**2.78E+08**
Worst	3.94E+08	2.02E++09	9.08E+08	3.80E+09	2.08E+08	1.01E+09	8.76E+07	4.48E+08
Mean	1.18E+08	1.16E+09	8.80E+08	3.63E+09	1.94E+08	9.19E+08	**6.80E+07**	**3.67E+08**
Std	9.67E+07	5.96E+08	1.76E+07	9.58E+07	7.68E+06	4.83E+07	**7.57E+06**	**4.45E+07**
Rank	2	2	4	4	3	3	**1**	**1**
f7								
Best	8.14E+05	4.01E+06	3.59E+06	1.49E+07	1.55E+06	8.07E+06	**7.69E+05**	**4.50E+06**
Worst	3.08E+06	1.49E+24	1.18E+20	3.10E+22	2.00E+06	9.18E+06	9.55E+05	5.51E+06
Mean	2.26E+06	9.53E+06	3.83E+06	1.56E+07	1.80E+06	8.57E+06	**8.78E+05**	**4.87E+06**
Std	7.79E+05	3.20E+06	8.74E+04	2.71E+05	1.12E+05	3.15E+05	**5.38E+04**	**2.07E+05**
Rank	3	3	4	4	2	2	**1**	**1**
f8								
Best	4.19E+18	9.81E+21	2.64E+19	1.28E+22	5.24E+19	3.94E+22	**1.05E+18**	**9.72E+21**
Worst	5.36E+21	1.49E+24	1.18E+20	3.10E+22	2.67E+20	9.98E+22	1.25E+19	1.45E+22
Mean	2.62E+21	7.12E+23	6.15E+19	2.30E+22	1.86E+20	6.84E+22	**7.49E+18**	**1.20E+22**
Std	1.87E+21	5.39E+23	2.89E+19	6.04E+21	5.18E+19	1.69E+22	**2.52E+18**	**1.14E+21**
Rank	4	4	2	2	3	3	**1**	**1**

Table 21 (continued)

	BA		PSO		Jaya		IJaya	
	n = 500	n = 1000	n = 500	n = 1000	n = 500	n = 1000	n = 500	n = 1000
f9								
Best	2.34E−05	1.00E−02	1.80E+00	1.40E+00	2.58E−04	4.52E−04	**2.58E−06**	**6.77E−06**
Worst	2.63E−01	1.79E−01	1.68E+14	6.91E+14	1.57E−02	2.53E−02	1.58E−04	3.03E−04
Mean	6.67E−02	6.00E−02	2.88E+00	3.09E+00	3.17E−03	5.98E−03	**3.74E−05**	**7.68E−05**
Std	5.46E−02	4.80E−02	4.59E−01	7.43E−01	3.27E−03	5.83E−03	**3.72E−05**	**8.69E−05**
Rank	3	3	4	4	2	2	**1**	**1**
f10								
Best	1.39E+14	5.66E+14	1.56E+14	6.59E+14	7.09E++13	3.47E+14	**3.35E+13**	**1.92E+14**
Worst	1.63E+14	6.63E+14	1.68E+14	6.91E+14	8.86E+13	3.92E+14	4.16E+13	2.40E+14
Mean	1.50E+14	6.03E+14	1.64E+14	6.73E+14	7.75E+13	3.68E+14	**3.76E+13**	**2.09E+14**
Std	5.78E+12	2.69E+13	3.22E+12	8.69E+13	4.60E+12	1.22E+13	**2.40E+12**	**1.07E+13**
Rank	3	2	4	4	2	3	**1**	**1**
f11								
Best	− 9.26E+01	− 1.72E+02	− 7.35E+01	− 1.31E+02	**− 1.04E+02**	**− 1.98E+02**	− 9.86E+01	− 1.92E+02
Worst	2.26E+12	6.89E+13	9.82E+12	3.16E+14	− 9.13E+01	− 1.68E+02	− 9.28E+01	− 1.76E+02
Mean	− 7.92E+01	− 1.52E+02	− 6.55E+01	− 1.26E+02	**− 9.71E+01**	**− 1.84E+02**	− 9.52E+01	− 1.82E+02
Std	7.07E+00	7.41E+00	**2.63E+00**	2.58E+00	3.73E+00	7.43E+00	**1.88E+00**	3.18E+00
Rank	3	3	4	4	**1**	**1**	2	2
f12								
Best	8.35E+11	2.82E+13	8.26E+12	2.87E+14	**4.47E+10**	**7.59E+12**	2.26E+11	8.53E+12
Worst	2.26E+12	6.89E+13	9.82E+12	3.16E+14	3.66E+11	1.19E+13	3.49E+11	1.18E+13
Mean	1.37E+12	4.23E+13	9.26E++12	3.07E+14	**2.14E+11**	**9.29E+12**	2.69E+11	1.03E+13
Std	4.05E+11	9.18E+12	3.83E+11	7.28E+12	1.02E+11	1.22E+12	**3.20E+10**	**9.05E+11**
Rank	3	3	4	4	**1**	**1**	2	2

Table 21 (continued)

	BA		PSO		Jaya		IJaya	
	n = 500	n = 1000	n = 500	n = 1000	n = 500	n = 1000	n = 500	n = 1000
f13								
Best	1.41E+03	1.53E+03	5.36E+03	1.14E+04	1.43E+03	3.58E+03	**6.40E+02**	**1.60E+03**
Worst	3.31E+03	6.79E+03	6.08E+03	1.24E+04	2.02E+03	4.44E+03	1.05E+03	2.48E+03
Mean	2.39E+03	5.01E+03	5.76E+03	1.19E+04	1.70E+03	4.03E+03	**7.90E+02**	**2.15E+03**
Std	4.79E+02	1.21E+03	1.78E+02	2.88E+02	1.41E+02	2.13E+02	**1.08E+02**	**2.10E+02**
Rank	3	3	4	4	2	2	**1**	**1**
f14								
Best	− 2.78E+04	**− 5.75E+04**	− 4.47E+04	− 1.25E+04	− 8.36E+03	− 1.22E+04	**− 4.87E+04**	− 5.41E+04
Worst	8.33E+03	1.64E+04	9.01E+03	1.81E+04	1.28E+04	9.84E+03	− 1.56E+04	− 1.80E+04
Mean	− 2.45E+04	− 3.17E+04	− 2.05E+04	− 1.18E+04	− 5.45E+02	7.03E+01	**− 2.99E+04**	**− 3.19E+04**
Std	4.99E+03	5.36E+03	4.77E+03	2.84E+03	4.62E+03	6.99E+03	**4.60E+03**	**1.02E+03**
Rank	2	2	3	3	4	4	**1**	**1**
f15								
Best	5.62E+03	1.21E+04	8.48E+03	1.76E+04	6.08E+03	1.28E+04	**5.60E+03**	**1.16E+04**
Worst	8.33E+03	1.64E+04	9.01E+03	1.81E+04	7.37E+03	1.55E+04	6.03E+03	1.27E+04
Mean	7.08E+03	1.46E+04	8.77E+03	1.79E+04	6.66E+03	1.39E+04	**5.80E+03**	**1.21E+04**
Std	7.82E+02	2.25E+03	7.13E+02	2.64E+02	3.34E+02	5.92E+02	**1.23E+02**	**2.24E+02**
Rank	3	3	4	4	2	2	**1**	**1**
f16								
Best	1.98E+01	1.98E+01	2.00E+01	2.00E+01	2.00E+01	2.00E+01	**1.85E+01**	**1.91E+01**
Worst	1.28E+04	2.84E+04	2.00E+01	2.00E+01	2.07E+01	2.07E+01	1.91E+01	1.96E+01
Mean	2.04E+01	2.03E+01	2.00E+01	2.00E+01	2.02E+01	2.02E+01	**1.88E+01**	**1.94E+01**
Std	3.05E−01	2.47E− 01	**0**	**0**	3.05E−01	2.61E−01	1.76E−01	1.24E−01
Rank	4	4	2	2	3	3	**1**	**1**

Table 21 (continued)

	BA		PSO		Jaya		IJaya	
	n = 500	n = 1000	n = 500	n = 1000	n = 500	n = 1000	n = 500	n = 1000
f17								
Best	7.58E+03	1.26E+04	1.34E+04	2.78E+04	6.53E+03	1.46E+04	**3.33E+03**	**8.82E+03**
Worst	1.28E+04	2.84E+04	1.45E+04	2.90E+04	7.67E+03	1.68E+04	3.92E+03	1.04E+04
Mean	1.19E+04	2.46E+04	1.39E+04	2.84E+04	6.96E+03	1.57E+04	**3.65E+03**	**9.69E+03**
Std	1.04E+03	3.25E+03	2.64E+02	4.74E+02	3.23E+02	5.64E+02	**1.75E+02**	**4.31E+02**
Rank	3	3	4	4	2	2	**1**	**1**
f18								
Best	**1.14E+02**	**1.65E+02**	1.21E+02	1.76E+02	6.43E+02	9.09E+02	3.52E+02	4.90E+02
Worst	1.24E+02	1.75E+02	1.26E+02	1.79E+02	6.50E+02	9.20E+02	4.44E+02	5.90E+02
Mean	**1.20E+02**	**1.69E+02**	1.24E+02	1.78E+02	6.46E+02	9.14E+02	3.82E+02	5.42E+02
Std	2.71E+00	2.67E+00	**1.55E+00**	**8.73E−01**	1.78E+00	3.06E+00	1.76E+00	2.17E+00
Rank	**1**	**1**	2	2	4	4	3	3

Table 22 The statistical results of the BA and IJaya algorithms using Wilcoxon Signed Rank Test for dimensions of 10, 20, 30, 100, 500, and 1000

F	(IJaya—BA)(Pop = 25)											
	n = 10		n = 20		n = 30		n = 100		n = 500		n = 1000	
	P-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign
f1	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	0.0001033	(+)	0.0001399	(+)
f2	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f3	1	(−)	1	(−)	1	(−)	1	(−)	1	(−)	1	(−)
f4	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	0.0001401	(+)	0.000189	(+)
f5	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	0.3134631	(−)	0.000189	(+)
f6	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	0.0438037	(+)	0.0003902	(+)
f7	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	0.0001033	(+)	0.0006806	(+)
f8	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	0.0004493	(+)	0.0003902	(+)	0.0003902	(+)
f9	0.0001033	(+)	0.0001033	(+)	0.0001033	(+)	8.86E−05	(+)	0.0001033	(+)	0.0001033	(+)
f10	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f11	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f12	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f13	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	0.0001033	(+)
f14	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	0.0002932	(+)	0.0365613	(+)	0.0015073	(+)
f15	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	0.0002536	(+)	0.0001033	(+)
f16	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f17	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f18	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)

Table 23 The statistical results of the PSO and IJaya algorithms using Wilcoxon Signed Rank Test for dimensions of 10, 20, 30, 100, 500, and 1000

F	(IJaya—PSO)(Pop = 25)											
	n = 10		n = 20		n = 30		n = 100		n = 500		n = 1000	
	P-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign	*P*-value	Sign
f1	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.84E−05	(+)	8.84E−05	(+)	8.76E−05	
f2	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f3	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f4	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f5	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(++)	8.86E−05	(+)	8.86E−05	(+)
f6	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f7	8.86E−05	(++)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f8	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.84E−05	(+)
f9	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f10	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f11	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f12	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f13	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f14	8.86E−05	(+)	0.0001401	(+)	0.0002191	(+)	0.8812927	(+)	0.8812927	(+)	8.86E−05	(+)
f15	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f16	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f17	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)
f18	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)	8.86E−05	(+)

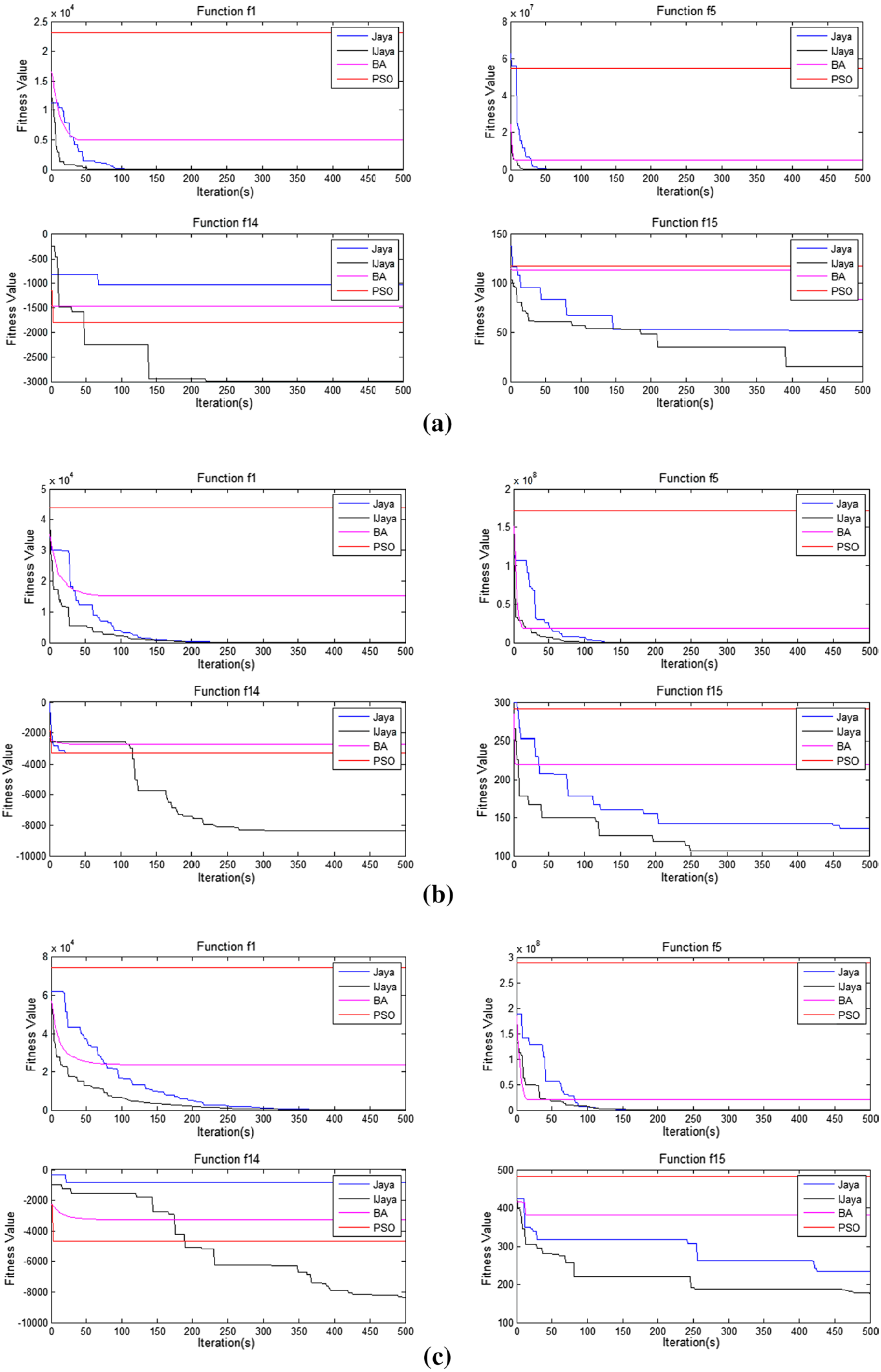


Fig. 7 Convergence graphs of the f1, f5, f14, and f15 functions for the BA, PSO, Jaya and IJaya algorithms **a** for dimension = 10, **b** for dimension = 20, **c** for dimension = 30, **d** for dimension = 100, **e** for dimension = 500, and **f** for dimension = 1000

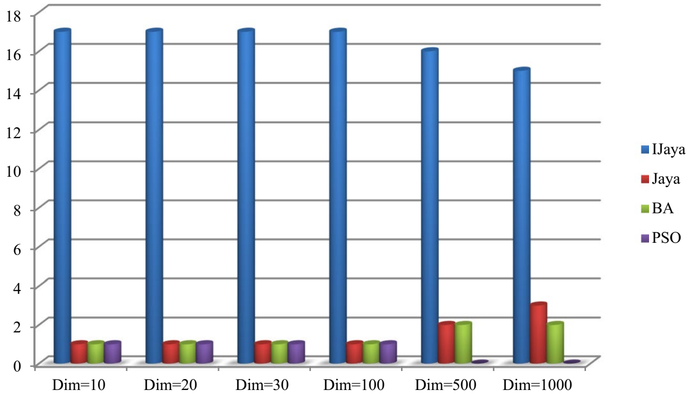


Fig. 8 Rank performances of Jaya, IJaya, BA, and PSO in terms of the mean results for the population size of 25 based on dimension sizes (Dim = dimension)

Table 24 The results of the IJaya and other algorithms for the dimension of 100

	PSO	CSS	GOA	SSA	MVO	Jaya	IJaya
f1							
Best	1.00E+04	5.20E+01	1.69E+04	4.90E−08	6.67E−02	1.76E−08	**5.02E−09**
Mean	4.87E+04	1.97E+04	1.77E+04	4.08E−08	6.77E−02	6.45E−08	**1.39E−08**
Std	2.11E+04	1.67E+04	5.96E+03	5.99E−09	1.35E−02	5.44E−08	**5.39E−09**
f2							
Best	–	–	–	–	–	1.27E+07	**6.68E−14**
Mean	–	–	–	–	–	1.68E+07	**3.80E−07**
Std	–	–	–	–	–	2.33E+06	**1.86E−06**
f3							
Best	6.44E+01	9.55E−03	3.49E+01	1.61E−22	8.68E−10	**0.00E+00**	**0.00E+00**
Mean	3.26E+02	3.02E+01	5.88E+01	1.43E−22	7.93E−10	**0.00E+00**	**0.00E+00**
Std	1.68E+02	2.37E+01	5.28E+01	4.60E−23	3.10E−10	**0.00E+00**	**0.00E+00**
f4							
Best	–	–	–	–	–	5.28E−02	**2.42E−03**
Mean	–	–	–	–	–	1.82E−01	**7.16E−03**
Std	–	–	–	–	–	9.56E−02	**5.00E−03**
f5							
Best	5.17E+02	5.02E+04	3.22E+06	9.35E+01	5.16E+02	2.74E+00	**4.89E−01**
Mean	1.70E+06	4.38E+05	6.65E+06	1.38E+02	3.86E+02	3.87E+01	**2.64E+00**
Std	1.23E+06	2.51E+05	3.32E+06	8.47E+01	4.75E+02	4.11E+01	**1.19E+00**
f6							
Best	–	–	–	–	–	9.81E+00	**3.33E+00**
Mean	–	–	–	–	–	1.36E+02	**1.87E+01**
Std	–	–	–	–	–	2.11E+02	**1.01E+01**
f7							
Best	1.32E+04	7.94E−01	9.33E+03	2.34E−02	2.97E+00	7.30E−09	**2.51E−09**
Mean	2.45E+04	6.95E+03	1.08E+04	1.46E−02	3.28E+00	3.59E−07	**5.78E−09**

Table 24 (continued)

	PSO	CSS	GOA	SSA	MVO	Jaya	IJaya
Std	6.63E+03	7.74E+03	2.70E+03	1.67E−02	3.38E+00	1.52E−06	**2.36E−09**
f8							
Best	–	–	–	–	–	4.63E+08	**5.83E+02**
Mean	–	–	–	–	–	6.78E+12	**6.72E+02**
Std	–	–	–	–	–	9.43E+12	**4.00E+01**
f9							
Best	1.00E+60	1.61E+33	3.05E+53	**1.03E−08**	4.14E−07	2.55E−05	2.85E−08
Mean	3.44E+79	3.34E+61	3.33E+68	**1.42E−08**	3.64E−07	4.97E+04	5.53E+00
Std	1.79E+80	1.80E+62	1.83E+69	**3.62E−09**	1.76E−07	1.25E+05	1.88E+01
f10							
Best	–	–	–	–	–	3.25E−01	**8.21E−02**
Mean	–	–	–	–	–	1.49E+00	**2.14E−01**
Std	–	–	–	–	–	1.15E+00	**6.78E−02**
f11							
Best	–	–	–	–	–	−3.11E+01	**−3.40E+01**
Mean	–	–	–	–	–	−2.59E+01	**−2.99E+01**
Std	–	–	–	–	–	1.67E+00	**1.45E+00**
f12							
Best	–	–	–	–	–	−1.83E+06	**−1.94E+07**
Mean	–	–	–	–	–	3.52E+07	**−9.24E+06**
Std	–	–	–	–	–	3.02E+07	**5.73E+06**
f13							
Best	4.57E+02	1.20E+02	1.11E+03	**4.21E+00**	9.68E+00	4.54E+01	5.00E+00
Mean	9.05E+02	4.68E+02	1.15E+03	1.41E+01	**1.09E+00**	6.72E+01	1.38E+00
Std	2.97E+02	1.93E+02	5.28E+02	5.76E+01	2.73E+00	1.20E+01	**2.40E+00**
f14							
Best	–	–	–	–	–	−4.63E+03	**−4.19E+04**
Mean	–	–	–	–	–	−6.55E+02	**−3.37E+04**
Std	–	–	–	–	–	1.75E+03	**1.03E+03**
f15							
Best	6.76E+02	2.37E+02	1.32E+03	3.54E+02	5.80E+02	4.38E+02	**2.17E+02**
Mean	8.56E+02	6.22E+02	1.14E+03	3.80E+02	5.08E+02	4.98E+02	**3.79E+02**
Std	8.42E+01	2.26E+02	1.72E+02	4.37E+01	7.18E+01	4.88E+01	**1.18E+01**
f16							
Best	1.93E+01	1.39E+01	2.00E+01	2.86E+00	1.96E+01	2.00E+01	1.83E−05
Mean	1.96E+01	1.56E+01	1.66E+01	3.17E+00	5.68E+00	2.00E+01	3.36E−01
Std	**1.77E−01**	1.04E+00	4.03E+00	6.39E−01	7.79E+00	7.57E−01	6.78E−01
f17							
Best	2.97E−01	4.96E−01	1.95E+02	6.41E−08	1.43E−01	9.80E−09	**3.65E−09**
Mean	3.99E+02	1.95E+02	1.67E+02	4.51E−03	1.49E−01	1.85E−03	**5.75E−04**
Std	1.93E+02	1.54E+02	5.31E+01	9.49E−03	2.94E−02	3.20E−03	**2.18E−03**
f18							
Best	–	–	–	–	–	4.04E+02	**1.45E+02**
Mean	–	–	–	–	–	4.10E+02	**1.93E+02**
Std	–	–	–	–	–	3.28E+01	**2.17E+01**

Table 25 Comparing Jaya and IJaya algorithms for population size = 20, 40, and 60 on various engineering design problems (MaxFEs = 30,000)

Problem	Jaya			IJaya		
	N = 20	N = 40	N = 60	N = 20	N = 40	N = 60
PVD						
Best	1.19E+04	1.20E+04	7.61E+03	1.90E+04	1.40E+04	1.01E+04
Worst	1.06E+05	7.42E+04	4.56E+04	1.39E+05	5.51E+04	4.27E+04
Mean	4.62E+04	3.43E+04	2.63E+04	**4.03E+04**	**2.78E+04**	**2.41E+04**
SD	2.21E+04	1.36E+04	9.05E+03	2.24E+04	9.37E+03	7.68E+03
WBD						
Best	4.16E+00	3.55E+00	3.03E+00	4.25E+00	3.69E+00	3.47E+00
Worst	1.66E+01	1.05E+01	1.24E+01	1.28E+01	8.32E+00	8.80E+00
Mean	**7.52E+00**	6.67E+00	**6.13E+00**	7.99E+00	**5.82E+00**	6.24E+00
SD	2.53E+00	1.71E+00	1.71E+00	1.93E+00	1.18E+00	1.34E+00
CSD						
Best	1.34E−02	1.29E−02	1.34E−02	2.48E−02	1.35E−02	1.73E−02
Worst	1.00E+08	1.00E+08	1.00E+08	1.00E+08	1.00E+08	1.00E+08
Mean	9.33E+07	5.33E+07	5.33E+07	**8.00E+07**	**5.00E+07**	**4.33E+07**
SD	2.49E+07	4.99E+07	4.99E+07	4.00E+07	5.00E+07	4.96E+07

Table 26 Comparing Jaya and IJaya algorithms for population size = 20, 40, and 60 on various engineering design problems (MaxFEs = 50,000)

Problem	Jaya			IJaya		
	N = 20	N = 40	N = 60	N = 20	N = 40	N = 60
PVD						
Best	1.21E+04	1.21E+04	8.87E+03	1.18E+04	1.13E+04	1.22E+04
Worst	8.01E+04	4.80E+04	5.58E+04	7.25E+04	6.34E+04	4.17E+04
Mean	4.08E+04	2.96E+04	2.70E+04	**4.06E+04**	**2.73E+04**	**2.54E+04**
SD	1.96E+04	1.05E+04	1.10E+04	1.35E+04	1.17E+04	7.46E+03
WBD						
Best	3.72E+00	2.96E+00	4.20E+00	3.21E+00	4.06E+00	3.37E+00
Worst	2.07E+01	1.03E+01	9.50E+00	2.09E+01	1.10E+01	8.81E+00
Mean	9.00E+00	**6.60E+00**	6.73E+00	**7.70E+00**	6.90E+00	**5.81E+00**
SD	4.23E+00	1.74E+00	1.55E+00	3.26E+00	1.69E+00	1.39E+00
CSD						
Best	1.33E–02	1.32E−02	2.23E−02	2.16E−02	1.75E−02	1.34E−02
Worst	1.00E+08	1.00E+08	1.00E+08	1.00E+08	1.00E+08	1.00E+08
Mean	9.00E+07	**5.33E+07**	5.33E+07	**7.00E+07**	6.67E+07	**5.00E+07**
SD	3.00E+07	4.99E+07	4.99E+07	4.58E+07	4.71E+07	5.00E+07

Table 27 Comparing Jaya and IJaya algorithms for population size = 20, 40, and 60 on various engineering design problems (MaxFEs = 100,000)

Problem	Jaya			IJaya		
	N = 20	N = 40	N = 60	N = 20	N = 40	N = 60
PVD						
Best	2.09E+04	9.25E+03	1.22E+04	1.56E+04	9.31E+03	1.00E+04
Worst	1.13E+05	7.51E+04	7.40E+04	6.29E+04	5.34E+04	5.30E+04
Mean	4.87E+04	3.25E+04	2.92E+04	**3.56E+04**	**2.58E+04**	**2.68E+04**
SD	2.03E+04	1.41E+04	1.13E+04	1.34E+04	1.12E+04	1.04E+04
WBD						
Best	3.35E+00	2.99E+00	2.82E+00	4.17E+00	4.17E+00	3.92E+00
Worst	1.45E+01	1.33E+01	9.44E+00	1.38E+01	9.59E+00	9.79E+00
Mean	**7.55E+00**	6.96E+00	6.34E+00	7.68E+00	**6.48E+00**	**6.34E+00**
SD	2.66E+00	2.32E+00	1.61E+00	2.07E+00	1.49E+00	1.34E+00
CSD						
Best	1.32E–02	2.89E–02	1.35E–02	1.33E–02	1.33E–02	1.34E–02
Worst	1.00E+08	1.00E+08	1.00E+08	1.00E+08	1.00E+08	1.00E+08
Mean	8.33E+07	5.67E+07	7.33E+07	**6.33E+07**	**5.67E+07**	**3.67E+07**
SD	3.73E+07	4.96E+07	4..42E+07	4.82E+07	4.96E+07	4.82E+07

Table 28 The results of the Wilcoxon Signed-Rank Test on the results of Jaya and IJaya algorithms on various engineering design problems

IJaya	Jaya (MaxFEs = 30,000)			Jaya (MaxFEs = 50,000)			Jaya (MaxFEs = 100,000)		
	N = 20	N = 40	N = 60	N = 20	N = 40	N = 60	N = 20	N = 40	N = 60
	P-value	*P*-value	*P*-value	*P*-value	*P*-value	*P*-value	*P*-value	*P*-value	*P*-value
PVD	0.2369	0.0449	0.2623	0.8936	0.1846	0.6143	0.0175	0.0256	0.5577
WBD	0.3600	0.0495	0.6143	0.1986	0.5999	0.0449	0.5716	0.4528	0.8774
CSD	0.1142	0.8854	0.8198	0.3982	0.2588	0.9093	0.2981	0.8464	0.0190

Table 29 Comparing IJaya and other algorithms on various engineering design problems

	f(x)	MaxFEs	Reference
PVD			
IJaya	**2.78E+04**	30,000	–
Jaya	3.43E+04	30,000	–
CTSA	6053.1729	30,000	Babalik et al. (2018)
GWO	6051.56390000	N/A	Mirjalili et al. (2014)
MFO	6059.71430000	N/A	Mirjalili (2015)
WOA	6059.74100000	6300	Mirjalili and Lewis (2016)
WBD			
IJaya	5.82E+00	30,000	–
Jaya	6.67E+00	30,000	–
CTSA	2.3901825	30,000	Babalik et al. (2018)
HS	**2.38070000**	110,000	Lee and Geem (2005)
Deb	2.43311600	5000	Deb (1991)

Author contributions EB: Conceptualization, ınvestigation, methodology, software, writing—review, original draft and editing.

Declarations

Conflict of interest The author declares that she has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Aslan M, Gunduz M, Kiran MS (2019) JayaX: Jaya algorithm with xor operator for binary optimization. Appl Soft Comput Journal 82:105576

Babalik A, Cinar AC, Kiran MS (2018) A modification of tree-seed algorithm using Deb's rules for constrained optimization. Appl Soft Comput 63:289–305

Baş E, Ülker E (2020) An efficient binary social spider algorithm for feature selection problem. Expert Syst Appl 146:113185

Baş E, Ülker E (2020) A binary social spider algorithm for uncapacitated facility location problem. Expert Syst Appl 161:113618

Baş E, Ülker E (2020a) A binary social spider algorithm for continuous optimization task. Soft Comput. https://doi.org/10.1007/s00500-020-04718-w

Baş E, Ülker E (2020c) Discrete social spider algorithm for the traveling salesman problem. Artif Intell Rev. https://doi.org/10.1007/s10462-020-09869-8

Baş E, Ülker E (2020e) Improved social spider algorithm for large scale constrained optimization. Artif Intell Rev. https://doi.org/10.1007/s10462-020-09931-5

Beşkirli M (2021) Solving continuous optimization problems using the tree seed algorithm developed with the roulette wheel strategy. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2021.114579

Caldeira RH, Gnanavelbabu A (2021) A Pareto based discrete Jaya algorithm for multi-objective flexible job-shop scheduling problem. Expert Syst Appl 170:114567

Chaudhuri A, Sahu TP (2021) A hybrid feature selection method based on binary Jaya algorithm for micro-array data classification. Comput Electr Eng 90:106963

Das H, Naik B, Behera HS, A Jaya algorithm based wrapper method for optimal feature selection in supervised classification. J King Saud Univ: Comput Inf Sci xxx (xxxx) xxx.

Deb K (1991) Optimal design of a welded beam via genetic algorithms. AIAA J 29(11):2013–2015

Deng H, Peng L, Zhang H, Yang B, Chen Z (2019) Ranking-based biased learning swarm optimizer for large-scale optimization. Inf Sci 493(2019):120–137

Deng W, Shang S, Cai X, Zhao H, Zhou Y, Chen H, Deng W (2021) Quantum differential evolution with cooperative coevolution framework and hybrid mutation strategy for large scale optimization. Knowl-Based Syst 224:107080

Elattar E, ElSayed SK (2019) Modified JAYA algorithm for optimal power flow incorporating renewable energy sources considering the cost, emission, power loss, and voltage profile improvement. Energy 178:598e609

Gao Y, Zhang H, Zhang Y (2019) Overlapping community detection based on conductance optimization in large-scale networks. Phys A 522(2019):69–79. https://doi.org/10.1016/j.physa.2019.01.142

Iacca G, Santos Junior VC, Melo VV (2021) An improved Jaya optimization algorithm with Lévy flight. Expert Syst Appl 165:113902

Kaveh A, Hosseini SM, Zaerreza A (2021) Improved Shuffled Jaya algorithm for sizing optimization of skeletal structures with discrete variables. Structures 29(2021):107–128

Lee KS, Geem ZW (2005) (2005), A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Methods Appl Mech Eng 194(36):3902–3933

Li Y, Zhao Y, Liu J (2021) Dynamic sine cosine algorithm for large-scale global optimization problems. Expert Syst Appl 177:114950

Li D, Guo W, Lerch A, Li Y, Wang L, Wu Q (2021) An adaptive particle swarm optimizer with decoupled exploration and exploitation for large scale optimization. Swarm Evol Comput 60:100789

Min W, Liu J, Zhang S (2018) Network-regularized sparse logistic regression models for clinical risk prediction and biomarker discovery. IEEE/ACM Trans Comput Biol Bioinform 15(2018):944–953. https://doi.org/10.1109/TCBB.2016.2640303

Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl-Based Syst 89(2015):228–249

Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95(2016):51–67

Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69(2014):46–61

Murty KG (2003) Optimization models for decision making, vol. 1, Chapter 1. pp. 1–18

Nayak DR, Zhang Y, Das DS, Panda S (2019) MJaya-ELM: A Jaya algorithm with mutation and extreme learning machine-based approach for sensorineural hearing loss detection. Appl Soft Comput J 83:105626

Rao RV, Keesari HS (2018) Multi-team perturbation guiding Jaya algorithm for optimization of wind farm layout. Appl Soft Comput 71(2018):800–815

Rao R (2016) Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. Int J Ind Eng Comput 7(1):19–34

Rather SA, Bala PK (2020) Swarm-based chaotic gravitational search algorithm for solving mechanical engineering design problems. World J Eng 17(1):97–114

Ren Z, Liang Y, Wang M, Yang Y, Chen A (2021) An eigenspace divide-and-conquer approach for large-scale optimization. Appl Soft Comput J 99:106911

Suganthan PN, Hansen N, Liang JJ, Deb K, Chen Y-P, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization. KanGAL Report 2005005:2005

Sun L, Lin L, Li H, Gen M (2019) Large scale flexible scheduling optimization by a distributed evolutionary algorithm. Comput Ind Eng 128(2019):894–904. https://doi.org/10.1016/j.cie.2018.09.025

Surjanovic S, Bingham D (2019) Virtual Library of simulation experiments: test functions and datasets, http://www.sfu.ca/ssurjano

Teng H, Chen Y, Zeng W, Shi Y, Hu Q (2010) A dual-system variable-grain cooperative coevolutionary algorithm: satellite-module layout design. IEEE Trans Evol Comput 14:438–455. https://doi.org/10.1109/TEVC.2009.2033585

Warid W (2020) Optimal power flow using the AMTPG-Jaya algorithm. Appl Soft Comput J 91:106252

Xinchao Z (2010) A perturbed particle swarm algorithm for numerical optimization. Appl Soft Comput 10(2010):119–124

Yang X-S (2011) Bat algorithm for multi-objective optimization. Int J Bio-Inspir Comput. https://doi.org/10.1504/IJBIC.2011.042259

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.