

Received 12 September 2022, accepted 23 September 2022, date of publication 26 September 2022, date of current version 5 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3210122

RESEARCH ARTICLE

Senior Learning JAYA With Powell's Method and Incremental Population Strategy

GÜRCAN YAVUZ 

Computer Engineering Department, Kutahya Dumlupinar University, 43100 Kutahya, Turkey
e-mail: gurcan.yavuz@dpu.edu.tr

ABSTRACT JAYA algorithm is one a recently developed meta-heuristic algorithm that does not require algorithm-specific parameters. It is an algorithm based on the fact that the solutions always go towards the best when searching. This paper proposes a JAYA variant (JAYA-SIP) with three improvements to the Original JAYA algorithm. It has incorporated the senior learning strategy, the incremental population strategy, and Powell's local search method into JAYA. The improvements were tested with IEEE Congress on Evolutionary Computation (CEC) benchmark set for 30 and 50 dimensions, and the benchmark functions set from a special issue of the Soft Computing journal (SOCO) for 500 and 1000 dimensions. In addition to benchmark sets, the performance of JAYA-SIP was evaluated with nine CEC 2011 real-world test functions. The results of the proposed algorithm are compared with JAYA variants and some meta-heuristic algorithms. According to the results of the experiment and the analysis, the proposed improvements increased the performance of the JAYA algorithm. JAYA-SIP achieved better results than the other algorithms it was compared with.

INDEX TERMS Benchmark suite, JAYA, Powell's method, optimization.


I. INTRODUCTION

Meta-heuristic algorithms are the leading methods used in solving real-world and engineering problems. Meta-heuristic algorithms, i.e. high-level methodologies, are not dependent on the problem being solved, making them applicable to a large number of problem types [1], [2]. The main reason for this is that when traditional methods are employed to solve a problem, the problem becomes more complex as the scale of the problem increases, necessitating a considerable amount of processing power and time [3], [4], [5], [6], [7], [8]. Meta-heuristic algorithms reduce the time it takes to identify the best solution or the solution that most suitable. This has led to an increased interest in algorithms and the development of a significant number of meta-heuristic algorithms over the past 10 years. Examples of these meta-heuristic algorithms that researchers are deeply interested in are Particle Swarm Optimization (PSO) [9], Ant Colony Optimization (ACO) [10], Artificial Bee Colony (ABC) [11],

Gravitational Search Algorithm (GSA) [12], Tree Seed Algorithm (TSA) [13], Firefly Algorithm (FFA) [14], Salp Swarm Algorithm (SSA) [15], [16], Differential Evolution (DE) [17], [18], [19], [20], and their variants [21]. JAYA is one of the latest meta-heuristic algorithms [22].

The JAYA algorithm has attracted considerable attention due to its simple structure and the absence of algorithm-dependent parameters. JAYA, by its very nature, has the ability to converge quickly. It does, however, have problem of getting stuck in the local optimum. To eliminate this shortcoming, researchers have made a number of changes to the algorithm.

In this study, a JAYA variant algorithm called JAYA-SIP for short, proposes three improvements in total, namely senior learning, incremental social learning, and Powell's local search method. These three improvements were implemented in order to prevent issues such as local optima and early convergence of the JAYA algorithm. The behavior of this algorithm in CEC 2014 and large-scale test functions was investigated and compared with the JAYA variant and current meta-heuristic algorithms.

The associate editor coordinating the review of this manuscript and approving it for publication was Mu-Yen Chen .

JAYA-SIP includes Powell's local search method and this method is one of the methods used by combining it with various meta-heuristic algorithms. For example, [23] tried to improve the exploitation behavior of the ABC algorithm by combining the ABC algorithm and Powell's and used it in 22 unconstrained and 13 constrained function solutions. In another study, the Grey Wolf Optimizer algorithm was extended by Powell's local search method, and the variant they name PGWO was used for data clustering [24]. [25] solved the minimum energy broadcast (MEB) problem in the wireless sensor network with a variant of the Flower Pollination Algorithm based on Powell's method.

Aydın, Yavuz, and Stützle 2017 proposed a generalized, configurable ABC framework with an ISL mechanism. In addition, two other studies proposed an ABC variant by including ISL in the ABC algorithm (Yavuz, Aydın, and Stützle 2016; Yavuz and Aydın 2019).

JAYA-SIP uses Incremental Social Learning (ISL) strategy as its incremental population strategy [26]. There are also a number of different meta-heuristic algorithms using this method in the literature. De Oca *et al.* included the ISL method in the PSO algorithm [27]. Liao *et al.* brought together the ACO algorithm and ISL [28]. Özyön, Yaşar, and Temurtaş included the ISL method in the gravitational search algorithm (GSA) and used it to solve high-dimensional problems [29]. Özyön and Aydın used IABC, a variant of Artificial Bee Colony Algorithm (ABC) with an ISL mechanism, to solve the economic power dispatch problem in the prohibited operating zone [30]. Aydın, Yavuz, and Stützle proposed a generalized, configurable ABC framework with an ISL mechanism [31]. In addition, two other studies proposed an ABC variant by including ISL in the ABC algorithm [32], [33].

The contributions of this work are summarized as follows:

- A JAYA variant known as JAYA-SIP is proposed.
- The senior learning mechanism is incorporated into the proposed algorithm.
- CEC 2014, SOCO, and Real World Problems are solved with JAYA-SIP.
- The ISL strategy and Powell's Method have been applied to the JAYA algorithm.

This study focuses on three proposed improvements for the Original JAYA algorithm. Section II briefly mentions the previous studies in the literature. Section III presents the general structure of the Original JAYA algorithm. In Section IV, the proposed improvements to the JAYA algorithm and the details of the proposed JAYA variant algorithm are given. Section V deals with the experimental environment, the results obtained by the algorithms included in the experiments, and their comparisons. Finally, Section VI summarizes the study and outlines the results.

II. RELATED WORKS

Many improvements have been made on the JAYA algorithm in the literature. These can be categorized as follows:

- Using learning methods to increase population diversity
- Improvement of the solution update equation
- Hybridization of the original JAYA algorithm with other techniques

In the first category, researchers used a variety of learning methods to diversify the population of the solution and prevent the algorithm getting stuck to local optima. To determine the Photovoltaic model parameters, Yu *et al.* used a JAYA variant with experience-based and chaotic elite learning methods [34]. Rao and Rai proposed a JAYA variant using the quasi-oppositional-based learning method to increase the JAYA algorithm's population diversity [35]. Wang and Huang [36] have integrated the elite opposition-based learning mechanism into the JAYA algorithm. This mechanism is based on identifying effective solutions close to the global optimum [36]. A study by X. Yang and Gong improved upon their proposed Enhanced JAYA (EJAYA) algorithm with a strategy called Generalized opposition-based learning [37]. Alawad and Abed-alguni proposed a variant using three mutation methods for position updating and Refraction Learning as an initialization method for solving discrete real-world problems [38].

The solution update equation is the most important factor influencing the power of the JAYA algorithm. Various alternatives to the JAYA solution update equation have been proposed in the second category. Ingle and Jatoth proposed a position update equation based on Levy Flight to prevent the JAYA algorithm from remaining stuck in the local optimum [39]. Leghari *et al.* added a weight parameter to the position update equation and sought to determine its value adaptively [40]. Luu and Nguyen combined the JAYA algorithm's solution update step with the DE operator [41]. Jian and Weng used a JAYA variant solution with chaotic map-based and multiple solution update equations to determine the parameters of photovoltaic cells [42]. X. Yang and Gong added the improved solution update equation to the EJAYA algorithm alongside the learning method [37]. Rao and Keesari created the MTPG-Jaya algorithm, which uses multiple teams to search the solution space [43]. These teams use various equations to search the same population. Farah and Belazi, however incorporated three new mutation strategies based on chaotic maps into the algorithm to improve JAYA's performance and tested them with 16 benchmark functions [44].

Aside from the improvements made in the first two categories, the third category of JAYA algorithm improvement is the combination of the JAYA algorithm with other techniques. For example, Gholami, Olfat, and Gholami combined the crow search algorithm, which excels at global search, and the JAYA algorithm, which excels at local search, and tested it on 20 benchmark functions [45]. Alotaibi combined the firefly and JAYA algorithms to prevent local optima from becoming stuck [46]. Goudos *et al.* combined the Grey Wolf and JAYA algorithms and applied them to two different antenna designs [47]. Xiong *et al.* used the Differential

Evolution algorithm in combination with JAYA to determine the parameters of the solid oxide fuel cell model [48]. Kaur, Sharma, and Mishra proposed the JAYA-Bat algorithm as a solution for reducing cognitive radio network power consumption [49]. Kumar and Yadav [50] combined the Teaching Learning Based Optimization algorithm with JAYA [50] and Azizi *et al.* brought together the ant lion optimizer algorithm with JAYA [51]. Tefek and Beşkirli combined the JAYA algorithm with a method known as Elite Local Search, without affecting the JAYA algorithm's general structure, and used it to solve optimization problems [52]. Gupta, Kumar, and Srivastava combined JAYA with Powell's method and proposed three JAYA variants for solving the optimal power flow problem with a distributed generating unit [53]. Aslan, Gunduz, and Kiran hybridized their JAYA variant algorithm with a local search module and used it to solve binary optimization problems [54].

Although the JAYA algorithm was introduced as recently as 2016, it has already been used extensively and been applied to various types of problems [55], [56], [57]. Premkumar *et al.* developed a variant of JAYA that uses a chaotic mapping to determine photovoltaic cell parameters [58]. Luu and Nguyen proposed a new variant called Modified JAYA to identify solar cell parameters [41]. Gunduz and Aslan created a variation dubbed DJAYA, which they utilized to solve the discrete problem type traveling salesman problem by making two adjustments to JAYA [3]. Aslan, Gunduz, and Kiran developed a JAYA variant for binary optimization by combining the JAYA algorithm with the logic exclusive or operator [54]. Liu *et al.* proposed a JAYA variant for short-term forecasting of wind speed with the support vector machine (SVM) based JAYA-SVM and tested the success of their algorithm with seven different methods [59]. Warid developed the adaptive multiple teams perturbation-guiding Jaya (AMTPG-Jaya) for single goal optimum power flow (OPF) forms [60]. Degertekin, Lamberti, and Ugur proposed the Discrete Advanced JAYA (DAJA) algorithm for the optimization of truss structures under stress and displacement constraints, which are a discrete optimization type problem that is difficult to solve [61]. Rao and Keesari optimized the wind farm layout problems with the JAYA algorithm variant they developed [43]. Rao and Saroj minimized the total annual cost problem by using a variant referred to as Elitist-JAYA in the shell-and-tube heat exchangers design [62]. Thirumoorthy and Muneeswaran used a hybrid JAYA variant to solve the text clustering problem and compared it with some known meta-heuristic algorithms [63]. Chaudhuri and Sahu proposed a hybrid filter-wrapper approach based on JAYA and tested it on 10 micro-array datasets [64].

III. THE ORIGINAL JAYA ALGORITHM

The Original JAYA is a population-based meta-heuristic algorithm created by Venkata Rao in 2016 [22]. It takes its name from the Sanskrit word JAYA, which means victory. The algorithm has a straightforward structure and only requires general control parameters (the number of function calls (FES)

and population size(NP)), rather than algorithm-specific control parameters.

Like other population-based algorithms, the Original JAYA has a population of randomly-distributed solutions in the search space. The basic logic of the algorithm is based on moving a solution in the population away from the worst solution and approximating it to the best solution in the population. The pseudo-code of the Original JAYA is given in Algorithm 1.

The Original JAYA algorithm begins its execution by generating random solutions. The positions of all solutions in the population are updated by Equation 1 until the execution budget is completed. Thus, it is ensured that all solutions avoid the worst solution ($X_{wo,j}^t$) and reach the current best solution ($X_{best,j}^t$).

$$X_{i,j}^{t+1} = X_{i,j}^t + r1(X_{best,j}^t - |X_{i,j}^t|) - r2(X_{wo,j}^t - |X_{i,j}^t|) \quad (1)$$

In the equation, $X_{i,j}^t$ represents the j . dimension of the i . solution in t . iteration. $X_{best,j}^t$ and $X_{wo,j}^t$ show the dimension of the best and worst solution in the t . iteration, respectively. Finally, $r1$ and $r2$ are randomly selected numbers from the range of [0, 1].

Algorithm 1 The Original JAYA Algorithm

- 1: Determine population size (NP)
 - 2: Populate the population $P(i = 1, 2, \dots, n)$ with random solutions
 - 3: $t \leftarrow 1$ ▷ initialize iteration
 - 4: **while** The termination criteria are not met **do**
 - 5: Find the current best and worst solutions of the population
 - 6: **for** $i = 1$ to NP **do**
 - 7: Update the position of the current solution with Equation 1 using the information of the best and worst solutions.
 - 8: $i \leftarrow i + 1$
 - 9: $t \leftarrow t + 1$ ▷ increment iteration
-

IV. SENIOR LEARNING JAYA WITH POWELL'S METHOD AND INCREMENTAL POPULATION STRATEGY

The proposed algorithm and its components will be described in detail in this section.

A. SENIOR LEARNING STRATEGY

The exploration and exploitation capabilities of the Original JAYA algorithm are based on its position update Equation (Equation 1) which is described in Section III. Although JAYA has a powerful position update equation, it can at times be insufficient in complex problem types [52], [65]. In order to eliminate this deficiency, the JAYA-SIP algorithm has included a learning method called senior learning in the JAYA algorithm. In this approach, Equation 2 was used to replace JAYA's position update equation, and an algorithmic

component called Senior Pool (SE) was added to the algorithm as a result of this equation.

$$X_{i,j}^{t+1} = X_{i,j}^t + r1(X_{best,j}^t - |X_{i,j}^t|) - r2(SE_{ra,j} - |X_{i,j}^t|) \quad (2)$$

In the equation, $X_{i,j}^t$ represents the j . dimension of the i . solution in t . iteration. $X_{best,j}^t$ and $SE_{ra,j}$ show the j . dimension of the best solution and randomly selected solution from SE pool in the t . iteration, respectively. Finally, $r1$ and $r2$ are randomly selected numbers from the range of $[0, 1]$.

In JAYA, the positions of the solutions are updated with Equation 1 during iterations. As a result of each update operation, if the objective function value of the new solution is better than the objective value of the existing solution, the current solution is updated with the new solution, and the information of the old solution is discarded. In the original JAYA this could lead to the loss of previous knowledge and experience in the scanned region in the search space, and thus, these regions to be searched again in the subsequent iterations. In this proposed study, after the current solution's position update, if the objective value of the new solution is better than the objective value of the old solution, the old solution is replaced with the new solution and the information of the old solution is not discarded. Instead, this solution is labeled "Senior" and saved in a "Senior Pool (SE)". Thus, this pool preserves the previous search space experience.

Solutions are added to the senior pool throughout the iterations, and as a result, the pool size can reach enormous sizes. This causes bad solutions to fill the pool and causes the algorithm to waste execution time for bad solutions. To avoid this bad situation, the SE pool size is reduced at a certain rate in each iteration. For this purpose, a parameter called *ratio* ($ratio = 0.10$) is used, and the new pool size is determined using this parameter. After the determination process, the solutions placed in the pool are ranked according to their objective function values. Solutions with the worst objective function value are discarded/removed from the pool until the size of the SE reaches the newly calculated pool size. This prevents the SE from growing too large and keeping bad solutions in the pool. In addition, a minimum value (Minimum Pool Size, MPS) is defined for the pool size ($MPS = 3$). The size of the SE pool cannot fall below this value. Thus, it is ensured that the algorithm always works with a small number of Senior solutions.

B. POWELL'S METHOD

In order to strengthen the exploration ability of the JAYA-SIP algorithm, it is considered in this study to add a local search method to the JAYA algorithm. For this purpose, a well-known, hybridized with many meta-heuristic algorithms [23], [24], [25], using Brent's technique [66] Powell's conjugate directions method was preferred [67]. The key reason for this is that it provides better performance. In addition, it is easy to hybridize with the meta-heuristic algorithm.

Powell's method is not an approach called for in every iteration in the proposed algorithm. The aim is to keep

the execution budget from being spent. According to Algorithm 2, JAYA-SIP calls Powell's method when it thinks that the search process has entered stagnation. This is determined by a parameter specified as sg . The value of sg is initially set at 0. When the position of a solution is updated (Equation 2), if the objective function value of the new solution is worse than the current solution, the value of sg is increased by one. When the value of this sg reaches the "stagnation factor value" specified at the beginning of the algorithm, JAYA-SIP calls Powell's local search method, and the sg value is reset. The solution with the current best objective value is used as the starting point for Powell's local search method.

C. INCREMENTAL SOCIAL LEARNING STRATEGY

The final improvement of the proposed algorithm is the incremental population method. For this method, the Incremental Social Learning mechanism (ISL) has been selected [27]. ISL is tried to increase the diversity of the population by adding new solutions to the existing population with ISL and to avoid getting stuck in the local optimum. The number of the growth period, which is selected at the start of the algorithm and called *growthperiod*, determines how long it takes to add a new solution to the population. After each *growthperiod* iteration, a new solution is added to the population. Unlike the random generation method used during the initialization step, the new solution is created using Equation 3. The new solution is produced close to the best solution in the population. The logic here is that there could be superior solutions around the best solution. The ISL process continues until the maximum number of populations (NP_{max}).

$$\hat{X}_{i,j}^{t+1} = X_{best,j}^t + ra(0, 1) * (X_{best,j}^t - ra(X^l - X^u)) \quad (3)$$

$X_{best,j}^t$ represents the j . dimension of the best solution of the current population. X^l and X^u represents the minimum and maximum bounds of the solved problem. $ra(0, 1)$ represents a random number generated between 0 and 1. $\hat{X}_{i,j}^{t+1}$ is the j . dimension of the new solution generated.

V. EXPERIMENTS

The experimental environment and conditions designed to test the performance of the proposed JAYA-SIP algorithm will be explained in this section, followed by the outcomes of the algorithms in the experiments.

A. THE EXPERIMENT ENVIRONMENT

The JAYA-SIP algorithm's performance was evaluated by solving CEC 2014 [68], the large-scale SOCO benchmark suite [69], and real-world problems [70]. The results of JAYA-SIP were compared with Comprehensive Learning Jaya (CLJAYA) [71], JAYA [22], Improved Jaya algorithm with Levy flight (LJA) [65], Improved JA based on the Linearly Decreasing Weight (LW-IJAYA) [40], Modified JAYA (MJA) [41], Grey Wolf Optimizer (GWO) [72], Sine Cosine Algorithm (SCA) [73] and the Tree-Seed Algorithm (TSA) [13].

The first experiments conducted are the CEC 2014 benchmark set experiments. CEC 2014 is a benchmark set that

Algorithm 2 Pseudo-Code of JAYA-SIP

```

1: Determine population size (NP)
2: Set the problems dimensions (D)
3:  $t \leftarrow 1, flag_i \leftarrow 0$ 
4:  $stagnation, ratio \leftarrow 0.10, MPS \leftarrow 3$ 
5: Populate the population  $P(i = 1, 2, \dots, n)$  with random solutions
6:  $SE \leftarrow \emptyset$   $\triangleright$  Create the Senior pool (SE)
7: while  $FES \leq MaxFES$  do
8:   for  $i = 1$  to  $NP$  do
9:     for  $j = 1$  to  $D$  do
10:      Update  $X_{i,j}^t$  solution with Equation 2
11:      if  $f(X_{i,j}^{t+1}) < f(X_{i,j}^t)$  then
12:         $SE \leftarrow X_{i,j}^t$   $\triangleright$  Put the old solution in the pool
13:         $flag_i \leftarrow 0$ 
14:         $sg \leftarrow 0$ 
15:      else
16:         $flag_i \leftarrow flag_i + 1$ 
17:         $sg \leftarrow sg + 1$ 
18:       $i \leftarrow i + 1$ 
19:    if  $sg < stagnation$  then
20:      Determine the current best solution of the population
21:      Use the current best solution of the population as the starting point and apply Powell's method
22:       $sg \leftarrow 0$ 
23:      Apply the increasing population method (ISL)
24:      if  $t > 0$  and  $t \% growthperiod == 0$  and  $NP < NP_{max}$  then
25:         $P \leftarrow \hat{X}_{i,j}^{t+1}$ 
26:         $poolsize \leftarrow NP \times ratio$   $\triangleright$  set pool size
27:        Sort solutions in Senior Pool(SE) by objective function values
28:        Discard the worst solutions from the pool until the senior pool size equals  $poolsize$ .
29:         $t \leftarrow t + 1$ 

```

has become a standard for evaluating the performance of optimization algorithms and includes functions of various difficulty levels. This benchmark set has 30 different numerical minimization test functions, including unimodal ($f1 - f3$), multimodal ($f4 - f16$), hybrid ($f17 - f22$), and composite ($f23 - f30$) types. The test functions in the benchmark set are summarized in Table 2. The search space of the functions in the table is defined in the range of $[-100, 100]$. The functions in this benchmark set are resolved for 30 and 50 dimensions. CEC 2014 experiments were carried out by adhering to the competition rules specified in the CEC 2014 competition. According to a study, the execution budget of all algorithms participating in the competition is $D \times 10000$ number of function calls (FES) [68]. D is the size of the problem. The algorithms were run 51 times independently for all functions. The obtained mean error value (where $(f(x^*) - f(x), f(x)$,

is the objective function value obtained by the algorithm, and $f(x^*)$ is the global optimum value of the test function) is used in the comparison. The error values obtained by following the rules of [68] were accepted as 10^{-8} when they were less than 10^{-8} .

The second part of the experiments was carried out for large-scale test functions. For this purpose, the SOCO benchmark set prepared for the Large Scale Optimization special issue of Soft Computing Journal was used. This benchmark set can be seen in Table 3. This benchmark set includes 19 functions, 4 separable and 15 non-separable SOCO experiments were carried out following the operating rules specified in [69]. Accordingly, the algorithms were run independently 25 times for each test function. The execution budget is equal to the number of $D \times 5000$ function calls. The algorithms were compared by taking the median value of the obtained error values ($f(x) - f(x^*)$). The error values obtained by following the rules of [69] were accepted as 10^{-14} when they were less than 10^{-14} .

The performance of the JAYA-SIP algorithm was tested using real-world engineering problems in the third part of the experiments. Nine test functions were selected from the CEC 2011 benchmark set. The experiments were carried out in accordance with the conditions specified in [70]. The JAYA-SIP results were compared with the JAYA variants.

All of the CEC 2014, SOCO and Real World problem experiments were conducted using an Ubuntu Linux computer with an AMD Ryzen 5 1600 with 16GB of RAM.

In order to conduct the experiments fairly, the control parameters of JAYA-SIP and the algorithms included in the experiments were configured using the irace tool [74], a parameter configuration tool. For parameter configuration, CEC 2014 10 dimension is used as a training set. The execution budget was selected as $D \times 5000$. The parameters of the algorithms, parameter range values, and parameter values configured with irace are listed in Table 1.

B. EXAMINING THE EFFECTS OF THE PROPOSED MODIFICATIONS

This section examines the contributions of the proposed improvements to the Original JAYA. For this, each improvement has been added to the JAYA algorithm separately and run on the CEC 2014 benchmark set for 30 and 50 dimensions. The proposed algorithm (JAYA-SIP) has been compared with JAYA+ISL, which is the ISL added version of JAYA, JAYA+Powell, which is Powell's method added to JAYA, and JAYA+SL, which is the inclusion of the Senior learning method

The mean error values obtained through the algorithms are presented in Table 4 for 30 dimensions and in Table 5 for 50 dimensions. At the bottom of the tables, the results of the pairwise comparison between the improvements with JAYA-SIP are given. "Win" means JAYA-SIP algorithm wins, "Lost" means the compared improvement wins, and

TABLE 1. The algorithms in the experiments, the names of the control parameters they have, the parameter types, the interval values of the parameters and the tuned values determined by the irace are given.

| Algorithm name | Abbreviation | Year | Parameter name | Parameter Type | Range | Tuned value |
|---|--------------|------|----------------|----------------|------------|-------------|
| Senior Learning JAYA with Powell’s Method and Incremental Population Strategy | JAYA-SIP | 2021 | NP | Integer | [4, 500] | 69 |
| | | | maxfoodsize | Integer | [20, 300] | 262 |
| | | | growthperiod | Integer | [1, 20] | 4 |
| | | | stagnation | Integer | [1, 50] | 18 |
| | | | lsItr | Integer | [3, 100] | 93 |
| JAYA | JAYA | 2016 | NP | Integer | [4, 500] | 16 |
| Comprehensive learning Jaya algorithm | cljaya | 2020 | NP | Integer | [4, 500] | 70 |
| Improved Jaya optimization algorithm with Lévy flight | lja | 2020 | NP | Integer | [4, 500] | 15 |
| | | | beta | Real | [1.0, 2.0] | 1.57 |
| | | | NP | Integer | [4, 500] | 119 |
| Improved JA based on the linearly decreasing weight parameter | Lw-ijaya | 2020 | ω_{Max} | Real | [0.4, 1.0] | 0.96 |
| | | | ω_{Min} | Real | [0.0, 0.4] | 0.31 |
| | | | NP | Integer | [4, 500] | 461 |
| Modified JAYA | mja | 2020 | NP | Integer | [4, 500] | 9 |
| Tree-Seed algorithm | TSA | 2015 | NP | Integer | [4, 500] | 9 |
| | | | ST | Real | [0.1, 0.5] | 0.38 |
| Grey Wolf Optimizer | GWO | 2014 | NP | Integer | [4, 500] | 84 |
| Sine Cosine Algorithm | SCA | 2016 | NP | Integer | [4, 500] | 8 |

TABLE 2. CEC 2014 benchmark suite.

| Type | Id | Functions |
|--------------------------------|-----|---|
| Unimodal | f01 | Rotated high conditioned elliptic function |
| | f02 | Rotated bent cigar function |
| | f03 | Rotated discus function |
| Multimodal | f04 | Shifted and rotated Rosenbrock function |
| | f05 | Shifted and rotated Ackley’s function |
| | f06 | Shifted and rotated Weierstrass function |
| | f07 | Shifted and rotated Griewan’s function |
| | f08 | Shifted Rastrigin function |
| | f09 | Shifted and rotated Rastrigin’s function |
| | f10 | Shifted Schwefel function |
| | f11 | Shifted and rotated Schwefel’s function |
| | f12 | Shifted and rotated Katsuura function |
| | f13 | Shifted and rotated HappyCat function |
| | f14 | Shifted and rotated HGBat function |
| | f15 | Shifted and rotated Expanded Griewank’s plus Rosenbroc’s function |
| | f16 | Shifted and rotated Expanded Scaffer’s F6 function |
| Hybrid | f17 | Hybrid function 1 (f9,f8,f1,f9,f8,f1) |
| | f18 | Hybrid function 2 (f2,f12,f8,f2,f12,f8) |
| | f19 | Hybrid function 3 (f7,f6,f4,f14,f7,f6,f4,f14) |
| | f20 | Hybrid function 4 (f12,f3,f13,f8,f12,f3,f13,f8) |
| | f21 | Hybrid function 5 (f14,f12,f4,f9,f1,f14,f12,f4,f9,f1) |
| | f22 | Hybrid function 6 (f10,f11,f13,f9,f5,f10,f11,f13,f9,f5) |
| Composition | f23 | Composition function 1 (f4,f1,f2,f3,f1,f4,f1,f2,f3,f1) |
| | f24 | Composition function 2 (f10,f9,f14,f10,f9,f14) |
| | f25 | Composition function 3 (f11,f9,f1,f11,f9,f1) |
| | f26 | Composition function 4 (f11,f13,f1,f6,f7,f11,f13,f1,f6,f7) |
| | f27 | Composition function 5 (f14,f9,f11,f6,f1,f14,f9,f11,f6,f1) |
| | f28 | Composition function 6 (f15,f13,f11,f16,f1,f15,f13,f11,f16,f1) |
| | f29 | Composition function 7 (f17,f18,f19,f17,f18,f19) |
| | f30 | Composition function 8 (f20,f21,f22,f20,f21,f22) |
| Search range : $[-100, 100]^D$ | | |

“Draw” means the comparison is a draw. The Rank row shows the average ranking value obtained by the algorithms in 30 problems.

First, JAYA-SIP and improvements on JAYA were compared for 30 dimensions in Table 4. Accordingly, it appears that JAYA-SIP, which contains all the improvements in its

structure, showed great success. While JAYA-SIP achieved lower error values in 28 of 30 problems against JAYA+ISL, JAYA+ISL achieved smaller values in 2 of them. In addition, the suggested method surpassed JAYA+SL in 25 problems while falling short in 5 others. When compared with another improvement, local search integration, JAYA+Powell, it is

TABLE 3. SOCO benchmark suite.

| Id | Name | Range | Seperable | Unimodal(U)/ Multimodal(M) |
|-----|---------------------------------|-------------------|-----------|----------------------------|
| f1 | Shifted Sphere Function | [-100, 100] | Yes | U |
| f2 | Shifted Schwefel's Problem 2.21 | [-100, 100] | No | U |
| f3 | Shifted Rosenbrock's Function | [-100, 100] | No | M |
| f4 | Shifted Rastrigin's Function | [-5, 5] | Yes | M |
| f5 | Shifted Griewank's Function | [-600, 600] | No | M |
| f6 | Shifted Ackley's Function | [-32, 32] | Yes | M |
| f7 | Schwefel's Problem 2.22 | [-10,10] | Yes | U |
| f8 | Schwefel's Problem 1.2 | [-65.536, 65.536] | No | U |
| f9 | Extended f10 | [-100, 100] | No | U |
| f10 | Bohachevsky | [-15, 15] | No | U |
| f11 | Schaffer | [-100, 100] | No | U |
| f12 | Hybrid f9 & f1 (25%, 75%) | [-100, 100] | No | M |
| f13 | Hybrid f9 & f3 (25%, 75%) | [-100, 100] | No | M |
| f14 | Hybrid f9 & f4 (25%, 75%) | [-5, 5] | No | M |
| f15 | Hybrid f10 & f7 (25%, 75%) | [-10,10] | No | M |
| f16 | Hybrid f9 & f1 (50%, 50%) | [-100, 100] | No | M |
| f17 | Hybrid f9 & f3 (75%, 25%) | [-100, 100] | No | M |
| f18 | Hybrid f9 & f4 (75%, 25%) | [-5, 5] | No | M |
| f19 | Hybrid f10 & f7 (75%, 25%) | [-10,10] | No | M |

TABLE 4. CEC 2014 30-dim results of proposed improvements over the Original JAYA. (The best results are highlighted in bold.)

| Functions | JAYA-SIP | JAYA+ISL | JAYA+SL | JAYA+Powell |
|-----------|-------------------|-------------------|-------------------|-------------------|
| f01 | 4.1671E+02 | 2.8001E+08 | 1.9772E+07 | 1.1832E+02 |
| f02 | 1.0000E-08 | 3.3485E+10 | 3.5272E+07 | 1.0000E-08 |
| f03 | 1.0000E-08 | 9.8822E+04 | 4.0266E+02 | 1.0000E-08 |
| f04 | 2.8589E+00 | 3.7705E+03 | 5.4024E+01 | 2.5645E+00 |
| f05 | 2.0003E+01 | 2.0913E+01 | 2.0167E+01 | 2.0041E+01 |
| f06 | 2.1042E+01 | 3.5982E+01 | 2.1056E+01 | 2.2196E+01 |
| f07 | 4.6682E-02 | 2.8572E+02 | 5.3905E-01 | 3.9054E-02 |
| f08 | 1.0000E-08 | 2.6651E+02 | 9.3104E+01 | 1.0000E-08 |
| f09 | 1.2803E+02 | 3.1905E+02 | 1.2872E+02 | 1.5712E+02 |
| f10 | 1.0868E+02 | 6.0430E+03 | 2.4152E+03 | 1.2340E+02 |
| f11 | 3.1804E+03 | 6.7895E+03 | 3.3814E+03 | 3.2640E+03 |
| f12 | 2.0440E-01 | 2.4256E+00 | 1.9287E-01 | 2.3697E-01 |
| f13 | 3.2798E-01 | 4.1813E+00 | 5.6685E-01 | 4.0589E-01 |
| f14 | 4.0077E-01 | 8.0812E+01 | 9.3311E-01 | 4.7999E-01 |
| f15 | 1.1422E+02 | 4.8744E+05 | 1.8596E+02 | 4.2589E+02 |
| f16 | 1.1280E+01 | 1.2788E+01 | 1.1495E+01 | 1.1595E+01 |
| f17 | 8.4245E+03 | 6.0992E+06 | 1.9171E+06 | 1.5425E+04 |
| f18 | 1.1412E+02 | 1.6889E+08 | 7.5660E+03 | 1.4406E+02 |
| f19 | 1.3481E+01 | 1.5106E+02 | 1.6201E+01 | 1.5092E+01 |
| f20 | 6.6206E+03 | 3.1117E+04 | 4.7339E+02 | 7.1800E+03 |
| f21 | 2.7631E+04 | 1.3195E+06 | 1.3635E+05 | 3.9766E+04 |
| f22 | 7.8962E+02 | 6.5896E+02 | 5.6138E+02 | 8.5148E+02 |
| f23 | 3.1524E+02 | 4.8264E+02 | 3.1683E+02 | 3.1524E+02 |
| f24 | 2.3666E+02 | 3.5781E+02 | 2.4108E+02 | 2.3939E+02 |
| f25 | 2.1589E+02 | 2.3187E+02 | 2.0872E+02 | 2.1027E+02 |
| f26 | 1.0237E+02 | 1.1047E+02 | 1.0486E+02 | 1.0241E+02 |
| f27 | 7.7841E+02 | 1.1247E+03 | 7.4031E+02 | 9.1956E+02 |
| f28 | 1.4214E+03 | 1.3987E+03 | 1.7041E+03 | 1.4115E+03 |
| f29 | 5.7635E+05 | 5.0833E+06 | 1.8776E+07 | 2.3385E+06 |
| f30 | 3.1526E+03 | 9.0410E+04 | 7.2509E+04 | 3.2434E+03 |
| rank | 1.40 | 3.80 | 2.60 | 2.10 |
| Our-win | | 28 | 25 | 22 |
| Our-lost | | 2 | 5 | 5 |
| draw | | 0 | 0 | 3 |

observed that JAYA-SIP had 22 wins and 5 losses and “draw” in 3 functions.

Afterwards, the effects of the proposed improvements on the JAYA algorithm when the size is increased were examined. Accordingly, when Table 5, which contains 50 dimension results, was analyzed, it was discovered that the order of the 30 dimension results remained constant as the dimension was increased. It can be said that JAYA-SIP achieved better results than the other algorithms. The proposed algorithm had 30 wins against JAYA+ISL, 26 wins against JAYA+SL, 4 losses, and finally, 20 wins, 8 losses, and 2 draws against JAYA+Powell.

Not considering the JAYA-SIP results, it can be said that Powell's local search method outperformed the others, only when the suggested improvements were examined among

TABLE 5. CEC 2014 50-dim results of proposed improvements over the original JAYA. (The best results are highlighted in bold.)

| Functions | JAYA-SIP | JAYA+ISL | JAYA+SL | JAYA+Powell |
|-----------|------------------|-----------|------------------|------------------|
| f01 | 1.098E+04 | 1.192E+09 | 5.005E+07 | 1.071E+04 |
| f02 | 1.000E-08 | 1.025E+11 | 5.198E+07 | 1.000E-08 |
| f03 | 2.280E-08 | 1.954E+05 | 6.806E+02 | 9.014E-08 |
| f04 | 2.911E+01 | 1.824E+04 | 9.411E+01 | 3.213E+01 |
| f05 | 2.001E+01 | 2.114E+01 | 2.017E+01 | 2.003E+01 |
| f06 | 4.054E+01 | 6.679E+01 | 4.332E+01 | 4.368E+01 |
| f07 | 5.149E-02 | 1.045E+03 | 9.408E-01 | 2.463E-02 |
| f08 | 1.000E-08 | 6.080E+02 | 2.244E+02 | 1.000E-08 |
| f09 | 2.658E+02 | 7.389E+02 | 2.955E+02 | 2.994E+02 |
| f10 | 1.805E+02 | 1.268E+04 | 4.664E+03 | 5.646E+02 |
| f11 | 6.070E+03 | 1.331E+04 | 7.399E+03 | 6.642E+03 |
| f12 | 2.147E-01 | 3.317E+00 | 2.605E-01 | 2.725E-01 |
| f13 | 3.722E-01 | 6.642E+00 | 6.467E-01 | 4.016E-01 |
| f14 | 4.244E-01 | 2.777E+02 | 4.166E-01 | 4.456E-01 |
| f15 | 4.670E+03 | 8.668E+06 | 5.451E+03 | 4.131E+03 |
| f16 | 1.990E+01 | 2.248E+01 | 2.044E+01 | 2.024E+01 |
| f17 | 2.022E+04 | 4.827E+07 | 6.793E+06 | 7.682E+03 |
| f18 | 2.520E+02 | 1.957E+09 | 3.074E+03 | 2.977E+02 |
| f19 | 1.844E+01 | 4.257E+02 | 3.634E+01 | 1.768E+01 |
| f20 | 7.657E+03 | 1.075E+05 | 1.039E+03 | 1.251E+04 |
| f21 | 1.015E+05 | 1.727E+07 | 3.241E+06 | 1.164E+05 |
| f22 | 1.839E+03 | 2.180E+03 | 1.435E+03 | 1.776E+03 |
| f23 | 3.440E+02 | 1.038E+03 | 3.461E+02 | 3.440E+02 |
| f24 | 2.689E+02 | 5.715E+02 | 2.889E+02 | 2.702E+02 |
| f25 | 2.253E+02 | 3.284E+02 | 2.134E+02 | 2.165E+02 |
| f26 | 1.026E+02 | 1.780E+02 | 1.590E+02 | 1.586E+02 |
| f27 | 1.537E+03 | 1.989E+03 | 1.708E+03 | 1.562E+03 |
| f28 | 2.628E+03 | 3.252E+03 | 3.577E+03 | 2.426E+03 |
| f29 | 1.317E+07 | 5.302E+07 | 1.589E+08 | 2.038E+07 |
| f30 | 9.995E+03 | 6.184E+05 | 5.701E+05 | 1.017E+04 |
| rank | 1.40 | 3.93 | 2.70 | 1.90 |
| Our-win | | 30 | 26 | 20 |
| Our-lost | | 0 | 4 | 8 |
| draw | | 0 | 0 | 2 |

themselves. However, no single improvement has achieved better results than JAYA-SIP.

C. CEC 2014 TEST RESULTS

CEC 2014 experiments were conducted in two phases. The performance of the JAYA-SIP algorithm was first compared with the Original JAYA algorithm and its improved current variants. After that, some meta-heuristic algorithms from the literature were compared. Experiments were conducted for 30 and 50 dimensions.

The proposed algorithm and the algorithms included in the experiments were compared in pairs with Wilcoxon's rank-sum test at the 0.05 significance level. Wilcoxon rank-sum test results of the algorithm compared with JAYA-SIP are given in Tables 7, 9, 11 and 13 for CEC 2014. The results of SOCO experiments are also given at the bottom of the tables. \approx sign is used when the results obtained are not significant, the + sign is used if JAYA-SIP is significantly better, and - if JAYA-SIP is significantly worse.

In addition, summary information of statistical test results is given in Draw, Win, and Lost lines below the tables. Moreover, the average ranking value obtained by the algorithms is also included in the rank row.

1) EXPERIMENTS WITH JAYA VARIANTS

The 30 and 50 dimension results of JAYA-SIP and JAYA variants are presented in Table 6 and Table 8. In the

TABLE 6. CEC 2014 30-dim results of JAYA-SIP and JAYA variants. (The best results are highlighted in bold.)

| Func. | proposed | | JAYA | | CLJAYA | | LJA | | LW-IJAYA | | MJA | |
|----------|------------------|----------|------------------|-----------|------------------|-----------|-----------|-----------|------------------|-----------|------------------|-----------|
| | mean | std | mean | std | mean | std | mean | std | mean | std | mean | std |
| f1 | 4.167E+02 | 2.27E+03 | 3.021E+06 | 3.835E+06 | 1.800E+06 | 1.126E+06 | 4.802E+07 | 6.681E+07 | 6.598E+07 | 1.943E+07 | 2.188E+08 | 1.121E+08 |
| f2 | 1.000E-08 | 1.53E-16 | 4.988E-03 | 1.146E-02 | 9.741E+01 | 1.052E+02 | 7.608E+08 | 1.566E+09 | 3.343E+09 | 7.240E+08 | 1.130E+10 | 5.554E+09 |
| f3 | 1.000E-08 | 5.72E-15 | 1.167E+03 | 3.296E+03 | 4.824E+03 | 3.189E+03 | 1.104E+04 | 1.572E+04 | 3.298E+04 | 6.869E+03 | 9.562E+04 | 3.825E+04 |
| f4 | 2.859E+00 | 1.34E+01 | 4.034E+01 | 3.534E+01 | 4.455E+01 | 4.430E+01 | 1.315E+02 | 9.300E+01 | 3.730E+02 | 5.316E+01 | 1.018E+03 | 5.280E+02 |
| f5 | 2.000E+01 | 8.78E-03 | 2.089E+01 | 6.783E-02 | 2.091E+01 | 5.384E-02 | 2.092E+01 | 5.674E-02 | 2.094E+01 | 5.555E-02 | 2.055E+01 | 1.121E-01 |
| f6 | 2.104E+01 | 3.12E+00 | 1.958E+01 | 4.447E+00 | 1.001E+01 | 3.853E+00 | 1.807E+01 | 4.246E+00 | 3.235E+01 | 2.157E+00 | 2.972E+01 | 3.528E+00 |
| f7 | 4.668E-02 | 4.95E-02 | 1.027E-01 | 5.450E-01 | 1.008E-02 | 1.270E-02 | 1.014E+01 | 2.286E+01 | 3.435E+01 | 5.428E+00 | 8.762E+01 | 4.506E+01 |
| f8 | 1.000E-08 | 0.00E+00 | 9.803E+01 | 3.039E+01 | 1.408E+02 | 3.844E+01 | 1.038E+02 | 4.150E+01 | 2.057E+02 | 1.637E+01 | 1.749E+02 | 3.875E+01 |
| f9 | 1.280E+02 | 3.26E+01 | 1.698E+02 | 6.876E+01 | 1.876E+02 | 1.543E+01 | 2.008E+02 | 2.535E+01 | 2.081E+02 | 1.370E+01 | 2.148E+02 | 3.736E+01 |
| f10 | 1.087E+02 | 2.12E+02 | 3.520E+03 | 1.714E+03 | 4.615E+03 | 9.919E+02 | 3.498E+03 | 1.288E+03 | 6.489E+03 | 3.380E+02 | 4.458E+03 | 8.556E+02 |
| f11 | 3.180E+03 | 4.81E+02 | 6.550E+03 | 8.393E+02 | 6.798E+03 | 3.521E+02 | 6.085E+03 | 6.241E+02 | 6.787E+03 | 3.008E+02 | 5.224E+03 | 6.530E+02 |
| f12 | 2.044E-01 | 7.65E-02 | 2.440E+00 | 2.966E-01 | 2.430E+00 | 2.773E-01 | 2.347E+00 | 3.222E-01 | 2.478E+00 | 2.145E-01 | 8.364E-01 | 2.901E-01 |
| f13 | 3.280E-01 | 1.04E-01 | 4.834E-01 | 1.173E-01 | 3.830E-01 | 7.166E-02 | 6.591E-01 | 5.405E-01 | 6.703E-01 | 8.427E-02 | 1.872E+00 | 8.808E-01 |
| f14 | 4.008E-01 | 1.57E-01 | 3.833E-01 | 1.958E-01 | 4.987E-01 | 2.218E-01 | 3.809E+00 | 1.037E+01 | 7.118E+00 | 4.313E+00 | 2.330E+01 | 1.425E+01 |
| f15 | 1.142E+02 | 3.35E+02 | 1.891E+02 | 4.699E+02 | 1.892E+02 | 4.609E+02 | 9.368E+02 | 2.376E+03 | 7.702E+02 | 2.204E+03 | 7.102E+04 | 1.374E+05 |
| f16 | 1.128E+01 | 8.49E-01 | 1.238E+01 | 4.641E-01 | 1.243E+01 | 2.607E-01 | 1.200E+01 | 3.327E-01 | 1.256E+01 | 2.588E-01 | 1.208E+01 | 4.432E-01 |
| f17 | 8.424E+03 | 2.47E+04 | 5.907E+05 | 5.738E+05 | 2.804E+05 | 1.966E+05 | 3.170E+06 | 5.896E+06 | 2.374E+06 | 1.047E+06 | 1.168E+07 | 9.436E+06 |
| f18 | 1.141E+02 | 3.62E+01 | 8.573E+03 | 7.831E+03 | 7.857E+03 | 8.870E+03 | 2.913E+07 | 1.591E+08 | 1.843E+07 | 1.069E+07 | 7.567E+07 | 1.126E+08 |
| f19 | 1.348E+01 | 1.58E+01 | 2.007E+01 | 2.366E+01 | 1.440E+01 | 2.022E+01 | 2.759E+01 | 3.279E+01 | 3.596E+01 | 1.355E+01 | 8.465E+01 | 4.737E+01 |
| f20 | 6.621E+03 | 8.04E+03 | 6.603E+02 | 5.746E+02 | 2.222E+03 | 1.166E+03 | 8.121E+03 | 1.597E+04 | 9.661E+03 | 5.325E+03 | 4.251E+04 | 2.604E+04 |
| f21 | 2.763E+04 | 3.22E+04 | 1.746E+05 | 1.932E+05 | 1.169E+05 | 1.439E+05 | 3.875E+05 | 3.831E+05 | 3.377E+05 | 3.442E+05 | 3.070E+06 | 3.427E+06 |
| f22 | 7.896E+02 | 2.47E+02 | 4.734E+02 | 2.175E+02 | 2.636E+02 | 1.638E+02 | 5.131E+02 | 2.268E+02 | 6.568E+02 | 1.514E+02 | 7.420E+02 | 2.162E+02 |
| f23 | 3.152E+02 | 1.24E-08 | 3.153E+02 | 3.024E-01 | 3.152E+02 | 5.515E-14 | 3.309E+02 | 2.557E+01 | 3.383E+02 | 5.426E+00 | 3.718E+02 | 3.555E+01 |
| f24 | 2.367E+02 | 7.17E+00 | 2.467E+02 | 8.672E+00 | 2.323E+02 | 7.553E+00 | 2.418E+02 | 1.073E+01 | 2.000E+02 | 2.356E-04 | 2.903E+02 | 2.293E+01 |
| f25 | 2.159E+02 | 1.05E+01 | 2.078E+02 | 4.698E+00 | 2.048E+02 | 2.144E+00 | 2.136E+02 | 8.045E+00 | 2.055E+02 | 5.171E+00 | 2.312E+02 | 1.134E+01 |
| f26 | 1.024E+02 | 1.38E+01 | 1.543E+02 | 9.186E+01 | 1.004E+02 | 8.472E-02 | 1.532E+02 | 8.941E+01 | 1.007E+02 | 1.171E-01 | 1.019E+02 | 9.904E-01 |
| f27 | 7.784E+02 | 2.56E+02 | 8.482E+02 | 2.188E+02 | 5.537E+02 | 1.349E+02 | 8.412E+02 | 1.936E+02 | 9.594E+02 | 2.243E+02 | 9.267E+02 | 2.181E+02 |
| f28 | 1.421E+03 | 4.21E+02 | 1.399E+03 | 2.869E+02 | 1.185E+03 | 2.852E+02 | 1.483E+03 | 3.533E+02 | 1.605E+03 | 3.484E+02 | 1.116E+03 | 1.294E+02 |
| f29 | 5.763E+05 | 1.97E+06 | 5.546E+06 | 6.305E+06 | 2.807E+06 | 4.738E+06 | 4.483E+06 | 5.101E+06 | 6.517E+06 | 7.836E+06 | 1.116E+06 | 3.237E+06 |
| f30 | 3.153E+03 | 1.13E+03 | 1.071E+04 | 2.744E+04 | 3.130E+03 | 1.296E+03 | 4.487E+04 | 4.968E+04 | 6.237E+04 | 4.090E+04 | 4.696E+04 | 5.141E+04 |
| rank | 1.87 | | 3.03 | | 2.53 | | 3.83 | | 4.87 | | 4.87 | |
| Win (+) | | | 21 | | 18 | | 23 | | 26 | | 27 | |
| Lost (-) | | | 4 | | 8 | | 2 | | 4 | | 2 | |
| Draw (≈) | | | 5 | | 4 | | 5 | | 0 | | 1 | |

TABLE 7. CEC 2014 30-dim Wilcoxon rank-sum test results of JAYA-SIP and JAYA variants.

| Func. | JAYA | CLJAYA | LJA | LW-IJAYA | MJA |
|-------|------|--------|-----|----------|-----|
| f1 | + | + | + | + | + |
| f2 | + | + | + | + | + |
| f3 | + | + | + | + | + |
| f4 | + | + | + | + | + |
| f5 | + | + | + | + | + |
| f6 | - | - | - | + | + |
| f7 | ≈ | - | ≈ | + | + |
| f8 | + | + | + | + | + |
| f9 | + | + | + | + | + |
| f10 | + | + | + | + | + |
| f11 | + | + | + | + | + |
| f12 | + | + | + | + | + |
| f13 | + | + | + | + | + |
| f14 | ≈ | + | + | + | + |
| f15 | + | ≈ | + | + | + |
| f16 | + | + | + | + | + |
| f17 | + | + | + | + | + |
| f18 | + | + | + | + | + |
| f19 | + | + | + | + | + |
| f20 | - | - | ≈ | + | + |
| f21 | + | + | + | + | + |
| f22 | - | - | - | - | ≈ |
| f23 | ≈ | ≈ | + | + | + |
| f24 | + | - | + | - | + |
| f25 | - | - | ≈ | - | + |
| f26 | + | ≈ | + | - | - |
| f27 | ≈ | ≈ | ≈ | + | + |
| f28 | ≈ | - | ≈ | + | - |
| f29 | + | + | + | + | + |
| f30 | + | ≈ | + | + | + |

table, the objective function error value of each function of 51 is given in the form of mean and standard deviation (std) metrics.

First, when the 30 dimension results are analyzed, JAYA-SIP was a better optimizer than the JAYA variants for all unimodal problems ($f1 - f3$). JAYA-SIP ranked first in 10 of the 13 Multimodal functions ($f4 - f16$), ranked second in two ($f7, f14$), and ranked fourth ($f6$) in one of them. In hybrid test functions, the proposed algorithm ranked first in four test functions, ahead of JAYA algorithms. As for $f20$ and $f22$ test functions, JAYA-SIP lagged behind JAYA variants. In two functions of composite functions ($f23 - f30$), it obtained the smallest error value ($f23, f29$), while it obtained the second smallest error value in two of them ($f27, f30$).

According to the 30 dimension results in Table 6, the proposed algorithm obtained smaller objective function values in 24 test functions from JAYA, 20 from CL-JAYA, 27 from LJA and MJA, and 26 from LW-IJAYA.

According to the 50 dimensions results of JAYA variants in Table 8, JAYA-SIP obtained the smallest error value in all unimodal functions. In multi-modal functions, the proposed algorithm was the best algorithm in 10 of the test functions. On the other hand, in hybrids, JAYA-SIP took first place in 4 functions. Finally, while it was the first algorithm in three of the composite functions, it came in second place in two others, resulting in competitive outcomes.

When the 50 dimension results in Table 8 are examined, the proposed JAYA-SIP algorithm achieved better or similar results than JAYA in 24 of 30 test functions, CLJAYA in 21 functions, LJA in 25 functions, and LW-IJAYA and MJA algorithms in 28 functions. However, the JAYA-SIP algorithm showed the best performance in

TABLE 8. CEC 2014 50-dim results of JAYA-SIP and JAYA variants. (The best results are highlighted in bold.)

| Func. | Proposed | | JAYA | | CLJAYA | | LJA | | LW-IJAYA | | MJA | |
|----------|------------------|----------|------------------|-----------|------------------|-----------|------------------|-----------|------------------|-----------|------------------|-----------|
| | mean | std | mean | std | mean | std | mean | std | mean | std | mean | std |
| f1 | 1.098E+04 | 1.43E+04 | 1.016E+07 | 3.819E+07 | 2.932E+06 | 1.212E+06 | 1.000E+08 | 1.060E+08 | 1.868E+08 | 4.336E+07 | 6.406E+08 | 2.458E+08 |
| f2 | 1.000E-08 | 1.41E-16 | 7.078E+03 | 8.515E+03 | 8.570E+03 | 9.123E+03 | 3.143E+09 | 4.914E+09 | 1.377E+10 | 1.920E+09 | 4.613E+10 | 1.389E+10 |
| f3 | 2.280E-08 | 1.10E-07 | 8.214E+03 | 1.090E+04 | 2.489E+04 | 6.910E+03 | 3.987E+04 | 3.613E+04 | 7.338E+04 | 8.878E+03 | 2.241E+05 | 6.078E+04 |
| f4 | 2.911E+01 | 5.28E+01 | 7.757E+01 | 3.181E+01 | 7.808E+01 | 3.299E+01 | 3.529E+02 | 6.835E+02 | 1.495E+03 | 2.979E+02 | 5.324E+03 | 3.197E+03 |
| f5 | 2.001E+01 | 2.06E-02 | 2.113E+01 | 3.945E-02 | 2.113E+01 | 2.980E-02 | 2.113E+01 | 4.184E-02 | 2.112E+01 | 4.176E-02 | 2.079E+01 | 1.228E-01 |
| f6 | 4.054E+01 | 4.32E+00 | 4.388E+01 | 5.695E+00 | 2.213E+01 | 6.181E+00 | 3.841E+01 | 6.946E+00 | 6.209E+01 | 3.141E+00 | 5.768E+01 | 3.393E+00 |
| f7 | 5.149E-02 | 8.32E-02 | 4.465E-01 | 1.409E+00 | 5.942E-03 | 7.881E-03 | 1.921E+01 | 4.037E+01 | 1.362E+02 | 2.342E+01 | 3.936E+02 | 1.362E+02 |
| f8 | 1.000E-08 | 0.00E+00 | 2.377E+02 | 5.045E+01 | 1.597E+02 | 1.170E+02 | 2.205E+02 | 8.430E+01 | 4.292E+02 | 1.948E+01 | 4.170E+02 | 4.770E+01 |
| f9 | 2.658E+02 | 4.69E+01 | 2.784E+02 | 7.023E+01 | 3.800E+02 | 2.823E+01 | 4.464E+02 | 7.188E+01 | 4.482E+02 | 2.018E+01 | 4.976E+02 | 5.189E+01 |
| f10 | 1.805E+02 | 3.53E+02 | 5.706E+03 | 2.041E+03 | 1.072E+04 | 1.821E+03 | 7.783E+03 | 2.315E+03 | 1.303E+04 | 3.901E+02 | 9.576E+03 | 1.433E+03 |
| f11 | 6.070E+03 | 8.23E+02 | 1.294E+04 | 5.712E+02 | 1.303E+04 | 4.971E+02 | 1.231E+04 | 6.816E+02 | 1.299E+04 | 4.464E+02 | 1.009E+04 | 8.815E+02 |
| f12 | 2.147E-01 | 6.15E-02 | 3.352E+00 | 2.605E-01 | 3.338E+00 | 3.056E-01 | 3.206E+00 | 3.336E-01 | 3.273E+00 | 2.930E-01 | 1.317E+00 | 3.420E-01 |
| f13 | 3.722E-01 | 8.87E-02 | 6.069E-01 | 1.077E-01 | 5.141E-01 | 9.896E-02 | 7.001E-01 | 3.868E-01 | 1.370E+00 | 5.235E-01 | 4.059E+00 | 7.262E-01 |
| f14 | 4.244E-01 | 1.04E-01 | 4.271E-01 | 2.444E-01 | 5.259E-01 | 2.607E-01 | 3.336E+00 | 9.298E+00 | 3.556E+01 | 6.128E+00 | 1.068E+02 | 4.017E+01 |
| f15 | 4.670E+03 | 1.05E+03 | 4.311E+03 | 1.177E+03 | 4.507E+03 | 1.005E+03 | 2.892E+04 | 8.280E+04 | 3.613E+04 | 4.205E+04 | 1.677E+06 | 1.738E+06 |
| f16 | 1.990E+01 | 8.23E-01 | 2.192E+01 | 4.737E-01 | 2.226E+01 | 2.733E-01 | 2.176E+01 | 4.223E-01 | 2.208E+01 | 2.350E-01 | 2.154E+01 | 4.973E-01 |
| f17 | 2.022E+04 | 9.24E+04 | 2.107E+06 | 1.762E+06 | 7.633E+05 | 4.340E+05 | 7.365E+06 | 6.122E+06 | 1.107E+07 | 3.296E+06 | 4.827E+07 | 3.224E+07 |
| f18 | 2.520E+02 | 7.40E+01 | 3.183E+03 | 2.072E+03 | 2.814E+03 | 2.246E+03 | 7.221E+07 | 1.551E+08 | 1.256E+08 | 4.196E+07 | 3.963E+08 | 3.647E+08 |
| f19 | 1.844E+01 | 2.61E+00 | 3.210E+01 | 2.360E+01 | 3.176E+01 | 1.615E+01 | 8.331E+01 | 8.480E+01 | 1.008E+02 | 1.921E+01 | 2.196E+02 | 7.505E+01 |
| f20 | 7.657E+03 | 6.62E+03 | 2.354E+03 | 6.294E+03 | 6.005E+03 | 4.549E+03 | 2.127E+04 | 1.946E+04 | 2.368E+04 | 8.301E+03 | 1.650E+05 | 1.620E+05 |
| f21 | 1.015E+05 | 8.92E+04 | 9.550E+05 | 8.641E+05 | 4.968E+05 | 3.024E+05 | 4.624E+06 | 5.346E+06 | 3.734E+06 | 1.276E+06 | 1.731E+07 | 1.099E+07 |
| f22 | 1.839E+03 | 4.21E+02 | 1.458E+03 | 3.673E+02 | 1.325E+03 | 4.685E+02 | 1.310E+03 | 3.230E+02 | 2.111E+03 | 3.685E+02 | 1.742E+03 | 3.954E+02 |
| f23 | 3.440E+02 | 5.82E-08 | 3.442E+02 | 1.287E+00 | 3.440E+02 | 7.075E-14 | 3.747E+02 | 4.880E+01 | 4.409E+02 | 1.771E+01 | 6.004E+02 | 1.174E+02 |
| f24 | 2.689E+02 | 6.47E+00 | 3.035E+02 | 1.111E+01 | 2.788E+02 | 4.099E+00 | 2.844E+02 | 1.340E+01 | 2.000E+02 | 1.115E-04 | 4.373E+02 | 3.174E+01 |
| f25 | 2.253E+02 | 1.29E+01 | 2.132E+02 | 7.178E+00 | 2.096E+02 | 2.960E+00 | 2.235E+02 | 1.003E+01 | 2.007E+02 | 4.807E+00 | 2.898E+02 | 2.635E+01 |
| f26 | 1.026E+02 | 1.38E+01 | 1.682E+02 | 1.138E+02 | 1.405E+02 | 8.908E+01 | 2.139E+02 | 1.218E+02 | 1.308E+02 | 4.470E+01 | 1.105E+02 | 4.631E+01 |
| f27 | 1.537E+03 | 1.36E+02 | 1.489E+03 | 1.198E+02 | 1.023E+03 | 1.280E+02 | 1.314E+03 | 1.323E+02 | 1.814E+03 | 1.027E+02 | 1.788E+03 | 9.832E+01 |
| f28 | 2.628E+03 | 8.33E+02 | 2.600E+03 | 6.315E+02 | 2.091E+03 | 4.708E+02 | 2.487E+03 | 5.142E+02 | 3.461E+03 | 7.817E+02 | 1.987E+03 | 4.327E+02 |
| f29 | 1.317E+07 | 1.73E+07 | 4.421E+07 | 3.361E+07 | 5.000E+07 | 3.158E+07 | 3.836E+07 | 2.944E+07 | 5.579E+07 | 5.340E+07 | 4.472E+07 | 1.540E+07 |
| f30 | 9.995E+03 | 2.04E+03 | 7.001E+04 | 1.512E+05 | 9.855E+03 | 3.886E+03 | 1.715E+05 | 2.807E+05 | 2.457E+05 | 1.570E+05 | 4.265E+05 | 2.788E+05 |
| rank | 1.8 | | 3.13 | | 2.77 | | 3.67 | | 4.77 | | 4.87 | |
| Win (+) | | | 21 | | 18 | | 24 | | 28 | | 28 | |
| Lost (-) | | | 4 | | 8 | | 2 | | 2 | | 1 | |
| Draw (≈) | | | 5 | | 4 | | 4 | | 0 | | 1 | |

$f1 - f5, f8 - f14, f16 - f19, f21, f23, f26, f29$ functions and took the first place.

In addition to the 30 and 50 dimension results, the JAYA-SIP algorithm's convergence speed was also investigated. From the CEC 2014 benchmark set, one unimodal, multimodal, composite, and hybrid test function was selected, and plots for 30 and 50 dimensions were created. The convergence curves obtained are shown in Figure 1. When the figures are compared, it is seen that the JAYA-SIP algorithm converges faster than other JAYA variants.

2) EXPERIMENTS WITH OTHER META-HEURISTICS

The results of JAYA-SIP are compared to some current meta-heuristic algorithms in this section. Controlled restart in differential evolution (b6e6rl)(The results of the b6e6rl algorithm are taken from the original article.) [75], TSA, GWO, and SCA were the meta-heuristic algorithms employed in the comparison. Table 10 shows the results of 30 dimensions. However, 50 dimension results are presented in Table 12.

Based on Table 10, it can be said that JAYA-SIP achieved better results than the other algorithms. While it obtained better results than TSA in 19 of the 30 functions, TSA which is one of the recent metaheuristic algorithms achieved better results in 9 of them, and the algorithms had similar results in 2 of them. However, it had 19 wins and 8 losses against GWO, which is one of the popular meta-heuristic algorithms, while the algorithms achieved the same results in 3 functions. While JAYA-SIP was ahead of SCA in 20 functions, it was behind in 9 of them. In one function, the result was a draw.

TABLE 9. CEC 2014 50-dim Wilcoxon rank-sum test results of JAYA-SIP and JAYA variants.

| Func. | JAYA | CLJAYA | LJA | LW-IJAYA | MJA |
|-------|------|--------|-----|----------|-----|
| f1 | + | + | + | + | + |
| f2 | + | + | + | + | + |
| f3 | + | + | + | + | + |
| f4 | + | + | + | + | + |
| f5 | + | + | + | + | + |
| f6 | + | - | ≈ | + | + |
| f7 | ≈ | - | + | + | + |
| f8 | + | + | + | + | + |
| f9 | ≈ | + | + | + | + |
| f10 | + | + | + | + | + |
| f11 | + | + | + | + | + |
| f12 | + | + | + | + | + |
| f13 | + | + | + | + | + |
| f14 | + | ≈ | + | + | + |
| f15 | - | - | ≈ | + | + |
| f16 | + | + | + | + | + |
| f17 | + | + | + | + | + |
| f18 | + | + | + | + | + |
| f19 | + | + | + | + | + |
| f20 | - | ≈ | + | + | + |
| f21 | + | + | + | + | + |
| f22 | - | - | - | + | ≈ |
| f23 | ≈ | ≈ | + | + | + |
| f24 | + | + | + | - | + |
| f25 | - | - | ≈ | - | + |
| f26 | + | ≈ | + | + | + |
| f27 | ≈ | - | - | + | + |
| f28 | ≈ | - | ≈ | + | - |
| f29 | + | + | + | + | + |
| f30 | + | ≈ | + | + | + |

The JAYA-SIP algorithm achieved 22 wins, 5 losses and 3 draws against b6e6rl.

TABLE 10. CEC 2014 30-dim results of some metaheuristic algorithms with JAYA-SIP. (The best results are highlighted in bold.)

| Func. | proposed | | TSA | | GWO | | SCA | | b6e6rl | |
|----------|------------------|----------|------------------|-----------|------------------|-----------|------------------|-----------|------------------|-----------|
| | mean | std | mean | std | mean | std | mean | std | mean | std |
| f1 | 4.167E+02 | 2.27E+03 | 1.040E+08 | 2.812E+07 | 3.590E+07 | 2.273E+07 | 3.999E+07 | 1.863E+07 | 4.054E+04 | 3.434E+04 |
| f2 | 1.000E-08 | 1.53E-16 | 3.547E+04 | 5.180E+04 | 1.369E+09 | 1.978E+09 | 1.036E+08 | 4.834E+07 | 1.000E-08 | 0.000E+00 |
| f3 | 1.000E-08 | 5.72E-15 | 3.080E+04 | 1.354E+04 | 1.137E+04 | 5.375E+03 | 3.780E+04 | 2.463E+04 | 1.000E-08 | 0.000E+00 |
| f4 | 2.859E+00 | 1.34E+01 | 1.208E+02 | 1.438E+01 | 1.363E+02 | 4.446E+01 | 1.273E+02 | 3.742E+01 | 1.620E+02 | 9.500E-02 |
| f5 | 2.000E+01 | 8.78E-03 | 2.095E+01 | 5.140E-02 | 2.095E+01 | 4.503E-02 | 2.094E+01 | 5.271E-02 | 2.027E+04 | 2.800E-02 |
| f6 | 2.104E+01 | 3.12E+00 | 3.029E+01 | 2.450E+00 | 1.359E+01 | 3.148E+00 | 3.724E+01 | 2.478E+00 | 1.213E+04 | 3.441E+00 |
| f7 | 4.668E-02 | 4.95E-02 | 1.023E-04 | 4.669E-04 | 1.126E+01 | 1.187E+01 | 2.526E+00 | 8.037E-01 | 1.000E-08 | 1.640E-09 |
| f8 | 1.000E-08 | 0.00E+00 | 1.424E+02 | 2.709E+01 | 6.587E+01 | 2.258E+01 | 8.871E+01 | 1.313E+01 | 1.000E-08 | 0.000E+00 |
| f9 | 1.280E+02 | 3.26E+01 | 2.074E+02 | 1.400E+01 | 9.017E+01 | 3.438E+01 | 1.328E+02 | 2.783E+01 | 4.410E+04 | 6.223E+00 |
| f10 | 1.087E+02 | 2.12E+02 | 5.173E+03 | 8.337E+02 | 2.469E+03 | 1.460E+03 | 4.439E+03 | 8.982E+02 | 3.500E+01 | 2.500E-02 |
| f11 | 3.180E+03 | 4.81E+02 | 7.094E+03 | 3.157E+02 | 4.927E+03 | 2.149E+03 | 5.952E+03 | 5.847E+02 | 1.980E+03 | 2.202E+02 |
| f12 | 2.044E-01 | 7.65E-02 | 2.625E+00 | 2.574E-01 | 2.446E+00 | 2.833E-01 | 2.459E+00 | 2.855E-01 | 3.390E+02 | 4.700E-02 |
| f13 | 3.280E-01 | 1.04E-01 | 5.058E-01 | 5.191E-02 | 4.173E-01 | 1.210E-01 | 4.693E-01 | 5.816E-02 | 3.360E+02 | 5.200E-02 |
| f14 | 4.008E-01 | 1.57E-01 | 3.405E-01 | 7.735E-02 | 2.067E+00 | 4.753E+00 | 2.796E-01 | 3.809E-02 | 2.430E+02 | 2.500E-02 |
| f15 | 1.142E+02 | 3.35E+02 | 3.097E+01 | 5.827E+00 | 1.240E+02 | 3.800E+02 | 2.574E+01 | 9.586E+00 | 5.808E+03 | 7.200E-01 |
| f16 | 1.128E+01 | 8.49E-01 | 1.295E+01 | 1.744E-01 | 1.142E+01 | 6.638E-01 | 1.343E+01 | 3.368E+01 | 9.539E+03 | 3.640E-01 |
| f17 | 8.424E+03 | 2.47E+04 | 2.661E+06 | 1.037E+06 | 8.974E+05 | 9.537E+05 | 1.638E+06 | 1.194E+06 | 2.116E+03 | 1.927E+03 |
| f18 | 1.141E+02 | 3.62E+01 | 8.817E+02 | 8.530E+02 | 1.921E+06 | 9.221E+06 | 1.171E+07 | 5.908E+06 | 2.870E+04 | 1.573E+01 |
| f19 | 1.348E+01 | 1.58E+01 | 8.856E+00 | 9.181E-01 | 1.733E+01 | 2.629E+00 | 2.076E+01 | 1.010E+00 | 4.386E+03 | 9.790E-01 |
| f20 | 6.621E+03 | 8.04E+03 | 1.307E+04 | 5.365E+03 | 4.695E+03 | 3.302E+03 | 6.521E+04 | 4.802E+04 | 1.538E+04 | 6.534E+00 |
| f21 | 2.763E+04 | 3.22E+04 | 5.333E+05 | 2.957E+05 | 3.646E+05 | 3.940E+05 | 8.291E+05 | 6.121E+05 | 2.431E+05 | 1.438E+02 |
| f22 | 7.896E+02 | 2.47E+02 | 4.822E+02 | 1.237E+02 | 4.032E+02 | 1.686E+02 | 3.868E+02 | 1.160E+02 | 6.362E+04 | 6.637E+01 |
| f23 | 3.152E+02 | 1.24E-08 | 3.152E+02 | 4.773E-06 | 3.274E+02 | 1.487E+01 | 3.223E+02 | 2.721E+01 | 3.152E+05 | 1.200E-09 |
| f24 | 2.367E+02 | 7.17E+00 | 2.258E+02 | 8.851E-01 | 2.071E+02 | 1.634E+01 | 2.000E+02 | 7.195E-03 | 2.229E+05 | 9.300E-01 |
| f25 | 2.159E+02 | 1.05E+01 | 2.233E+02 | 4.408E+00 | 2.024E+02 | 1.657E+00 | 2.028E+02 | 3.224E+00 | 2.033E+05 | 4.720E-01 |
| f26 | 1.024E+02 | 1.38E+01 | 1.005E+02 | 6.033E-02 | 1.044E+02 | 1.936E+01 | 1.005E+02 | 7.107E-02 | 1.004E+05 | 4.800E-02 |
| f27 | 7.784E+02 | 2.56E+02 | 6.903E+02 | 1.031E+02 | 7.797E+02 | 1.595E+02 | 1.062E+03 | 7.285E+01 | 3.516E+05 | 4.457E+01 |
| f28 | 1.421E+03 | 4.21E+02 | 9.362E+02 | 4.847E+01 | 4.430E+02 | 4.181E+01 | 4.554E+02 | 3.466E+01 | 8.205E+05 | 3.727E+01 |
| f29 | 5.763E+05 | 1.97E+06 | 2.659E+03 | 1.244E+03 | 2.139E+02 | 6.550E+00 | 2.074E+02 | 3.212E+00 | 7.868E+05 | 1.123E+02 |
| f30 | 3.153E+03 | 1.13E+03 | 4.736E+03 | 1.045E+03 | 4.506E+02 | 1.844E+02 | 4.105E+02 | 2.039E+02 | 1.318E+03 | 6.685E+02 |
| rank | 2.1 | | 3.23 | | 2.77 | | 3.1 | | 3.7 | |
| Win (+) | | | 19 | | 19 | | 20 | | 22 | |
| Lost (-) | | | 9 | | 8 | | 9 | | 5 | |
| Draw (≈) | | | 2 | | 3 | | 1 | | 3 | |

In Table 12, 50 dimension results are presented. Accordingly, JAYA-SIP won 22 of the 30 functions against TSA, lost 2 of them, and the result was a draw in 6 of them. Moreover, the proposed algorithm was found to be better in 19 of the 30 functions and worse in 9 of them than the GWO. When compared to SCA, it was seen that JAYA-SIP achieved better results in 20 functions, worse results in 8, and similar results in 2 of them. The JAYA-SIP algorithm achieved 21 wins, 8 losses and 1 draw against b6e6rl.

Based on the 30 and 50 dimension results in the tables, JAYA-SIP was the algorithm with the lowest average rank in 30 functions compared to other meta-heuristic algorithms. It was able to examine the search spaces of the functions in the benchmark set more effectively. This is due to the improvements in JAYA-SIP discussed earlier in this section, which strengthened the algorithm’s exploration and exploitation aspects.

D. SOCO LARGE SCALE EXPERIMENTS

In this section, the scalability behavior of the JAYA-SIP algorithm in large scale problems is examined. For this purpose, the SOCO large scale benchmark set was solved for 500 dimensions and 1000 dimensions. The median error values of the algorithms for 500 dimensions are presented in Table 14 and for 1000 dimensions in Table 15.

Wilcoxon’s rank-sum test was used to compare the proposed algorithm and other algorithms in pairs at the 0.05 significance level. Statistical results are given in the p-value line at the bottom of the tables. The signs ≈ are used if the results

TABLE 11. CEC 2014 30-dim Wilcoxon rank-sum test results of some metaheuristic algorithms with JAYA-SIP.

| Func. | TSA | GWO | SCA | b6e6rl |
|-------|-----|-----|-----|--------|
| f1 | + | + | + | + |
| f2 | + | + | + | ≈ |
| f3 | + | + | + | ≈ |
| f4 | + | + | + | + |
| f5 | + | + | + | + |
| f6 | + | - | + | + |
| f7 | - | + | + | - |
| f8 | + | + | + | ≈ |
| f9 | + | - | ≈ | + |
| f10 | + | + | + | - |
| f11 | + | + | + | - |
| f12 | + | + | + | + |
| f13 | + | + | + | + |
| f14 | - | ≈ | - | + |
| f15 | - | + | - | + |
| f16 | + | ≈ | + | + |
| f17 | + | + | + | - |
| f18 | + | + | + | + |
| f19 | ≈ | + | + | + |
| f20 | + | ≈ | + | + |
| f21 | + | + | + | + |
| f22 | - | - | - | + |
| f23 | ≈ | + | + | + |
| f24 | - | - | - | + |
| f25 | + | - | - | + |
| f26 | - | + | - | + |
| f27 | - | + | + | + |
| f28 | - | - | - | + |
| f29 | - | - | - | + |
| f30 | + | - | - | - |

of the algorithm compared with JAYA-SIP are not significant, + if the results of JAYA-SIP are significantly better, and - if

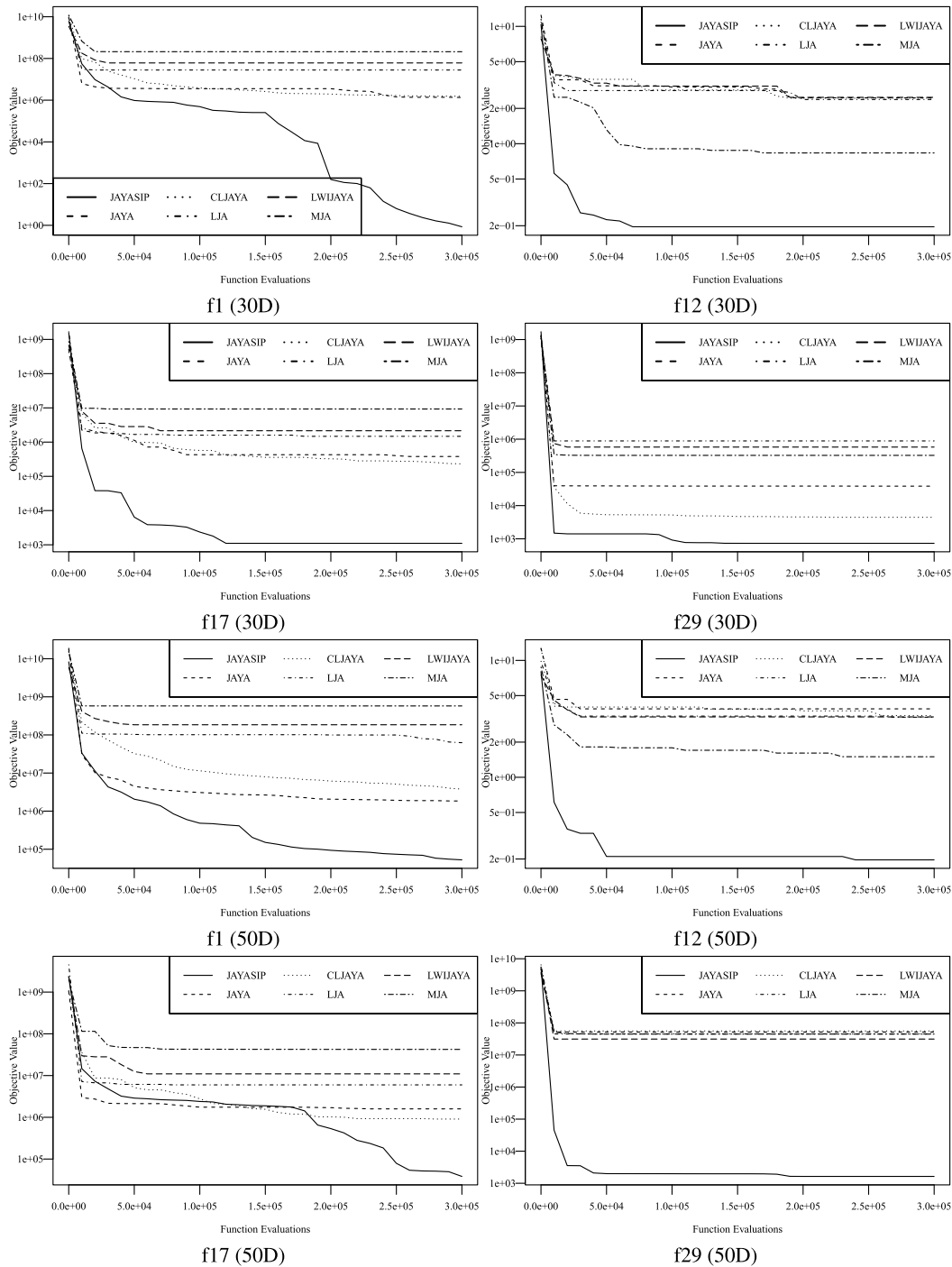


FIGURE 1. Convergence plots of JAYA variants for 30 and 50 dimensions. It is the median error value obtained by the algorithms after 51 runs. An example of Unimodal, Multimodal, Hybrid and Composite problem types is given.

the results of JAYA-SIP are worse. Furthermore, the numbers of wins, losses, and draws against the algorithm with which JAYA-SIP was compared are given in the Draw, Win, and Lost lines at the bottom of the tables.

When the 500 dimension results in Table 14 are examined, it is seen that the JAYA-SIP algorithm showed superior performance against the other algorithms it was compared to. JAYA-SIP won all 19 of the 19 functions of the SOCO

benchmark set against CLJAYA, LJA, LW-IJAYA, MJA, SCA and TSA. It only lost in one test function against GWO and JAYA. Considering the statistical tests performed, it can be said that the results of the proposed algorithm were significantly good.

When the problem size was increased to 1000, JAYA-SIP once again outperformed the other algorithms, as seen in Table 15. In the 1000 dimension median results, it was ahead

TABLE 12. CEC 2014 50-dim results of some metaheuristic algorithms with JAYA-SIP. (The best results are highlighted in bold.)

| Func. | proposed mean | std | TSA mean | std | GWO mean | std | SCA mean | std | b6e6rl mean | std |
|----------|------------------|----------|-------------|-----------|------------------|-----------|------------------|-----------|------------------|-----------|
| f1 | 1.098E+04 | 1.43E+04 | 3.586E+08 | 9.728E+07 | 7.885E+07 | 6.721E+07 | 1.047E+08 | 3.416E+07 | 3.711E+05 | 1.790E+05 |
| f2 | 1.000E-08 | 1.41E-16 | 2.095E+04 | 1.417E+04 | 7.274E+09 | 4.996E+09 | 1.127E+09 | 1.035E+09 | 5.012E+03 | 5.529E+03 |
| f3 | 2.280E-08 | 1.10E-07 | 1.341E+05 | 1.177E+04 | 2.871E+04 | 9.562E+03 | 1.133E+05 | 4.537E+04 | 2.386E+05 | 3.432E+02 |
| f4 | 2.911E+01 | 5.28E+01 | 1.008E+02 | 2.740E+00 | 6.248E+02 | 5.225E+02 | 3.013E+02 | 1.383E+02 | 4.946E+04 | 3.462E+01 |
| f5 | 2.001E+01 | 2.06E-02 | 2.115E+01 | 3.839E-02 | 2.114E+01 | 3.288E-02 | 2.112E+01 | 4.638E-02 | 2.037E+04 | 2.200E-02 |
| f6 | 4.054E+01 | 4.32E+00 | 6.249E+01 | 3.600E+00 | 3.093E+01 | 5.459E+00 | 7.125E+01 | 3.805E+00 | 2.749E+04 | 2.127E+00 |
| f7 | 5.149E-02 | 8.32E-02 | 1.678E-01 | 6.626E-02 | 5.418E+01 | 3.784E+01 | 8.928E+00 | 7.403E+00 | 1.740E-08 | 3.630E-09 |
| f8 | 1.000E-08 | 0.00E+00 | 3.169E+02 | 5.729E+01 | 1.598E+02 | 2.781E+01 | 2.258E+02 | 2.103E+01 | 1.000E-08 | 1.430E-09 |
| f9 | 2.658E+02 | 4.69E+01 | 4.390E+02 | 1.550E+01 | 1.852E+02 | 4.708E+01 | 3.026E+02 | 4.101E+01 | 1.021E+05 | 1.045E+01 |
| f10 | 1.805E+02 | 3.53E+02 | 1.074E+04 | 1.369E+03 | 4.923E+03 | 1.850E+03 | 1.022E+04 | 1.248E+03 | 3.500E+01 | 1.800E-02 |
| f11 | 6.070E+03 | 8.23E+02 | 1.357E+04 | 3.265E+02 | 1.082E+04 | 3.697E+03 | 1.202E+04 | 8.247E+02 | 4.395E+03 | 3.147E+02 |
| f12 | 2.147E-01 | 6.15E-02 | 3.547E+00 | 2.711E-01 | 3.247E+00 | 3.445E-01 | 3.327E+00 | 3.154E-01 | 3.490E+02 | 3.300E-02 |
| f13 | 3.722E-01 | 8.87E-02 | 6.746E-01 | 6.808E-02 | 7.693E-01 | 3.875E-01 | 6.543E-01 | 6.611E-02 | 4.690E+02 | 4.900E-02 |
| f14 | 4.244E-01 | 1.04E-01 | 4.567E-01 | 1.591E-01 | 9.065E+00 | 1.104E+01 | 3.829E-01 | 1.889E-01 | 2.870E+02 | 7.500E-02 |
| f15 | 4.670E+03 | 1.05E+03 | 4.978E+03 | 4.276E+02 | 6.188E+03 | 5.192E+03 | 4.684E+03 | 1.753E+03 | 1.263E+04 | 1.180E+00 |
| f16 | 1.990E+01 | 8.23E-01 | 2.266E+01 | 1.929E-01 | 2.105E+01 | 6.411E-01 | 2.296E+01 | 5.226E-01 | 1.777E+04 | 4.450E-01 |
| f17 | 2.022E+04 | 9.24E+04 | 2.010E+07 | 6.164E+06 | 4.442E+06 | 3.362E+06 | 9.843E+06 | 5.528E+06 | 1.753E+04 | 1.057E+04 |
| f18 | 2.520E+02 | 7.40E+01 | 1.085E+03 | 1.038E+03 | 1.270E+07 | 4.167E+07 | 6.567E+07 | 4.376E+07 | 1.229E+03 | 1.313E+03 |
| f19 | 1.844E+01 | 2.61E+00 | 5.445E+01 | 1.121E+01 | 5.473E+01 | 2.966E+01 | 4.218E+01 | 2.595E+00 | 1.330E+04 | 9.228E+00 |
| f20 | 7.657E+03 | 6.62E+03 | 5.067E+04 | 1.296E+04 | 8.366E+03 | 5.034E+03 | 5.102E+04 | 2.344E+04 | 6.284E+05 | 5.672E+02 |
| f21 | 1.015E+05 | 8.92E+04 | 8.819E+06 | 3.095E+06 | 1.562E+06 | 1.260E+06 | 3.519E+06 | 1.960E+06 | 2.409E+04 | 2.344E+04 |
| f22 | 1.839E+03 | 4.21E+02 | 1.631E+03 | 1.556E+02 | 1.092E+03 | 4.248E+02 | 1.055E+03 | 2.404E+02 | 5.689E+05 | 2.071E+02 |
| f23 | 3.440E+02 | 5.82E-08 | 3.440E+02 | 2.024E-05 | 3.945E+02 | 3.040E+01 | 3.139E+02 | 8.484E+01 | 3.440E+05 | 1.310E-09 |
| f24 | 2.689E+02 | 6.47E+00 | 2.670E+02 | 4.586E+00 | 2.380E+02 | 1.917E+01 | 2.033E+02 | 1.139E+01 | 2.620E+05 | 3.633E+00 |
| f25 | 2.253E+02 | 1.29E+01 | 2.711E+02 | 1.853E+01 | 2.038E+02 | 2.878E+00 | 2.020E+02 | 2.689E+00 | 2.071E+05 | 1.251E+00 |
| f26 | 1.026E+02 | 1.38E+01 | 1.007E+02 | 7.888E-02 | 1.104E+02 | 4.478E+01 | 1.006E+02 | 7.971E-02 | 1.181E+05 | 3.834E+01 |
| f27 | 1.537E+03 | 1.36E+02 | 1.718E+03 | 1.025E+02 | 1.305E+03 | 2.906E+02 | 1.763E+03 | 5.474E+01 | 9.011E+05 | 2.025E+02 |
| f28 | 2.628E+03 | 8.33E+02 | 1.420E+03 | 7.446E+01 | 5.237E+02 | 6.416E+01 | 5.287E+02 | 5.684E+01 | 1.236E+03 | 7.286E+01 |
| f29 | 1.317E+07 | 1.73E+07 | 1.312E+04 | 1.219E+04 | 2.245E+02 | 1.139E+01 | 2.050E+02 | 2.321E+00 | 1.317E+03 | 2.028E+02 |
| f30 | 9.995E+03 | 2.04E+03 | 1.006E+04 | 3.052E+03 | 1.082E+03 | 5.357E+02 | 5.477E+02 | 4.307E+02 | 9.605E+03 | 9.138E+02 |
| rank | 2.03 | | 3.6 | | 2.83 | | 2.77 | | 3.73 | |
| Win (+) | | | 22 | | 19 | | 20 | | 21 | |
| Lost (-) | | | 2 | | 9 | | 8 | | 8 | |
| Draw (≈) | | | 6 | | 2 | | 2 | | 1 | |

TABLE 13. CEC 2014 50-dim Wilcoxon rank-sum test results of some metaheuristic algorithms with JAYA-SIP.

| Func. | TSA | GWO | SCA | b6e6rl |
|-------|-----|-----|-----|--------|
| f1 | + | + | + | + |
| f2 | + | + | + | + |
| f3 | + | + | + | + |
| f4 | + | + | + | + |
| f5 | + | + | + | + |
| f6 | + | - | + | + |
| f7 | + | + | + | - |
| f8 | + | + | + | ≈ |
| f9 | + | - | + | + |
| f10 | + | + | + | - |
| f11 | + | + | + | - |
| f12 | + | + | + | + |
| f13 | + | + | + | + |
| f14 | ≈ | + | - | + |
| f15 | ≈ | + | ≈ | + |
| f16 | + | + | + | + |
| f17 | + | + | + | - |
| f18 | + | + | + | + |
| f19 | + | + | + | + |
| f20 | + | ≈ | + | + |
| f21 | + | + | + | - |
| f22 | - | - | - | + |
| f23 | + | + | ≈ | + |
| f24 | ≈ | - | - | + |
| f25 | + | - | - | + |
| f26 | ≈ | ≈ | - | + |
| f27 | + | - | + | + |
| f28 | - | - | - | - |
| f29 | ≈ | - | - | - |
| f30 | ≈ | - | - | - |

of CLJAYA, GWO, JAYA, LW-IJAYA and SCA algorithms in 18 test functions, and a function obtained JAYA-SIP worse

results from these algorithms. However, the proposed algorithm had 19 wins against LJA, MJA, and TSA.

SOCO test results showed that JAYA-SIP maintains its performance even as the problem size increases. This shows that the scalability aspect of the algorithm is strong.

E. REAL WORLD EXPERIMENTS

The proposed algorithm's performance has also been evaluated using real-world problems. Real-World Optimization Problems from the CEC 2011 Competition were preferred for this [70]. This benchmark set includes problems from Communication, Chemistry, Economics, and Astronomy. In the experiments, nine test functions from this benchmark set were used, and Table 16 provides a summary of their information. More detailed information about the problems can be found in [70].

Experiments with real-world problems were conducted by following the rules stated in [70]. Accordingly, each algorithm was independently run 25 times for every test function. Each algorithm is run up to the 150000 function evaluation (FES) and the parameters listed in Table 1 were used to run the algorithms. Table 17 displays the results, including the mean error value and standard deviation.

The results of the real world problems of the JAYA-SIP algorithm are compared with the JAYA variant algorithms. As a result, the proposed algorithm performed better than other JAYA variants and had an average ranking value of

TABLE 14. Median results of the algorithms for the SOCO 500 dimension. (The best results are highlighted in bold.)

| Func. | JAYA-SIP | CLJAYA | GWO | JAYA | LJA | LW-IJAYA | MJA | SCA | TSA |
|----------|------------------|-----------|-----------|------------------|-----------|-----------|------------|-----------|-----------|
| f1 | 1.000E-14 | 4.288E-10 | 8.258E+05 | 3.082E+03 | 2.831E+04 | 9.300E+05 | 1.821E+06 | 1.251E+05 | 1.201E+01 |
| f2 | 3.569E+01 | 1.194E+02 | 6.963E+01 | 1.331E+02 | 1.499E+02 | 9.601E+01 | 1.726E+02 | 1.250E+02 | 1.474E+02 |
| f3 | 1.331E-14 | 1.458E+03 | 2.503E+11 | 5.248E+08 | 1.042E+10 | 2.717E+11 | 1.525E+12 | 1.070E+09 | 2.679E+08 |
| f4 | 3.283E+01 | 2.919E+03 | 5.551E+03 | 4.671E+03 | 3.723E+03 | 7.905E+03 | 9.349E+03 | 6.196E+03 | 1.876E+03 |
| f5 | 1.000E-14 | 3.246E-01 | 6.294E+03 | 2.013E+01 | 2.434E+02 | 7.398E+03 | 1.547E+04 | 9.931E+02 | 9.863E-01 |
| f6 | 1.757E-11 | 1.691E+01 | 1.993E+01 | 1.956E+01 | 2.136E+01 | 2.063E+01 | 2.121E+01 | 2.124E+01 | 1.966E+01 |
| f7 | 4.780E-10 | 1.721E-07 | 8.368E+02 | 1.649E+01 | 6.473E+02 | 2.041E+47 | 2.539E+150 | 6.704E+57 | 3.999E-02 |
| f8 | 3.630E+05 | 5.416E+05 | 1.910E+05 | 1.607E+05 | 1.767E+06 | 6.613E+05 | 5.900E+06 | 4.851E+05 | 2.456E+06 |
| f9 | 5.069E+01 | 2.689E+03 | 3.301E+03 | 3.372E+03 | 5.013E+03 | 4.086E+03 | 5.292E+03 | 3.819E+03 | 3.258E+03 |
| f10 | 1.000E-14 | 1.757E+02 | 1.261E+04 | 1.463E+02 | 7.287E+02 | 1.405E+04 | 6.677E+04 | 2.608E+03 | 4.942E+01 |
| f11 | 6.488E+01 | 2.603E+03 | 3.326E+03 | 3.399E+03 | 4.907E+03 | 4.117E+03 | 5.331E+03 | 3.819E+03 | 3.245E+03 |
| f12 | 9.272E-04 | 1.291E+03 | 5.371E+05 | 2.697E+03 | 2.681E+04 | 5.971E+05 | 1.156E+06 | 2.528E+05 | 9.765E+02 |
| f13 | 8.460E-03 | 2.388E+03 | 1.388E+11 | 2.244E+08 | 1.821E+09 | 1.547E+11 | 9.799E+11 | 1.963E+08 | 1.127E+06 |
| f14 | 1.493E+01 | 2.042E+03 | 4.084E+03 | 3.589E+03 | 2.664E+03 | 6.000E+03 | 6.737E+03 | 4.920E+03 | 1.413E+03 |
| f15 | 4.140E-10 | 4.619E+01 | 2.369E+03 | 7.624E+01 | 5.601E+03 | 1.920E+16 | 5.751E+99 | 1.010E+13 | 7.978E+00 |
| f16 | 3.121E-03 | 2.472E+03 | 2.584E+05 | 3.273E+03 | 1.463E+04 | 2.941E+05 | 5.777E+05 | 2.699E+05 | 1.929E+03 |
| f17 | 2.243E+01 | 4.436E+03 | 2.348E+09 | 2.793E+06 | 8.775E+07 | 9.982E+09 | 6.698E+10 | 5.438E+08 | 3.447E+03 |
| f18 | 2.572E+01 | 9.691E+02 | 1.257E+03 | 1.633E+03 | 1.130E+03 | 2.339E+03 | 2.427E+03 | 1.894E+03 | 8.952E+02 |
| f19 | 2.144E-11 | 1.400E+02 | 5.870E+03 | 2.194E+02 | 1.804E+04 | 1.224E+04 | 5.921E+04 | 3.200E+03 | 2.659E+01 |
| rank | 1.11 | 2.79 | 5.58 | 4.21 | 5.95 | 7.32 | 8.89 | 6.11 | 3.05 |
| Our-win | | 19 | 18 | 18 | 19 | 19 | 19 | 19 | 19 |
| Our-lost | | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| draw | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| P-value | | 0.00000 | 0.00034 | 0.00065 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |

TABLE 15. Median results of the algorithms for the SOCO 1000 dimension. (The best results are highlighted in bold.)

| Func. | JAYA-SIP | CLJAYA | GWO | JAYA | LJA | LW-IJAYA | MJA | SCA | TSA |
|----------|------------------|-----------|-----------|------------------|-----------|------------|------------|------------|-----------|
| f1 | 1.000E-14 | 6.511E+02 | 2.198E+06 | 4.453E+04 | 1.040E+05 | 2.244E+06 | 4.268E+06 | 2.277E+05 | 2.693E+03 |
| f2 | 4.820E+01 | 1.296E+02 | 8.207E+01 | 1.384E+02 | 1.623E+02 | 9.804E+01 | 1.805E+02 | 1.521E+02 | 1.657E+02 |
| f3 | 1.000E-14 | 7.627E+07 | 7.280E+11 | 1.102E+10 | 3.463E+10 | 7.363E+11 | 3.731E+12 | 4.365E+09 | 2.063E+10 |
| f4 | 4.676E+01 | 7.698E+03 | 1.325E+04 | 9.589E+03 | 8.733E+03 | 1.655E+04 | 2.006E+04 | 1.342E+04 | 5.063E+03 |
| f5 | 1.000E-14 | 6.763E+00 | 1.945E+04 | 3.545E+02 | 9.158E+02 | 2.024E+04 | 3.916E+04 | 1.891E+03 | 4.460E+01 |
| f6 | 2.002E-11 | 1.930E+01 | 2.145E+01 | 1.966E+01 | 2.145E+01 | 2.092E+01 | 2.133E+01 | 2.124E+01 | 1.989E+01 |
| f7 | 1.760E-09 | 4.612E-02 | 2.362E+03 | 1.497E+02 | 1.437E+03 | 7.237E+140 | 3.211E+169 | 4.416E+168 | 2.602E+01 |
| f8 | 3.890E+06 | 1.516E+06 | 7.958E+05 | 6.050E+05 | 6.582E+06 | 1.936E+06 | 2.487E+07 | 2.064E+06 | 5.532E+06 |
| f9 | 2.396E+02 | 6.380E+03 | 7.458E+03 | 6.958E+03 | 1.027E+04 | 8.579E+03 | 1.101E+04 | 8.224E+03 | 7.507E+03 |
| f10 | 1.000E-14 | 3.160E+02 | 3.319E+04 | 2.019E+03 | 2.429E+03 | 3.358E+04 | 1.436E+05 | 5.405E+03 | 9.050E+02 |
| f11 | 2.711E+02 | 6.302E+03 | 7.524E+03 | 7.051E+03 | 1.058E+04 | 8.647E+03 | 1.100E+04 | 8.235E+03 | 7.474E+03 |
| f12 | 1.655E-03 | 3.100E+03 | 1.558E+06 | 1.778E+04 | 6.287E+04 | 1.534E+06 | 3.076E+06 | 4.675E+05 | 2.727E+03 |
| f13 | 6.279E+00 | 1.137E+07 | 4.880E+11 | 3.802E+09 | 1.858E+10 | 4.752E+11 | 2.823E+12 | 1.363E+09 | 1.468E+09 |
| f14 | 5.706E+01 | 5.668E+03 | 9.478E+03 | 7.317E+03 | 6.396E+03 | 1.228E+04 | 1.539E+04 | 1.038E+04 | 4.000E+03 |
| f15 | 8.877E-10 | 1.059E+02 | 5.914E+03 | 4.862E+02 | 1.512E+04 | 2.617E+75 | 4.052E+250 | 6.348E+81 | 6.358E+01 |
| f16 | 1.492E-02 | 5.415E+03 | 8.949E+05 | 1.157E+04 | 7.058E+04 | 8.321E+05 | 1.794E+06 | 4.954E+05 | 4.701E+03 |
| f17 | 1.840E+02 | 9.717E+03 | 3.789E+10 | 8.568E+07 | 1.810E+09 | 4.447E+10 | 4.206E+11 | 4.507E+07 | 8.179E+03 |
| f18 | 8.531E+01 | 2.387E+03 | 3.072E+03 | 3.354E+03 | 2.615E+03 | 4.615E+03 | 5.764E+03 | 4.210E+03 | 2.127E+03 |
| f19 | 1.427E-10 | 2.809E+02 | 1.526E+04 | 1.874E+03 | 4.441E+04 | 2.821E+04 | 1.899E+43 | 2.872E+04 | 4.777E+02 |
| rank | 1.26 | 2.53 | 6.00 | 4.05 | 5.95 | 6.89 | 8.89 | 5.95 | 3.47 |
| Our-win | | 18 | 18 | 18 | 19 | 18 | 19 | 18 | 19 |
| Our-lost | | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| draw | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| P-value | | 0.00079 | 0.00065 | 0.00065 | 0.00000 | 0.00034 | 0.00000 | 0.00042 | 0.00000 |

2.44 when compared to the other algorithms. As a result, the JAYA-SIP algorithm has shown the same performance in the real world benchmark set as in the CEC 2014 and SOCO.

F. ALGORITHM COMPLEXITY

The computational complexity analysis of the JAYA-SIP algorithm is performed in this section. The obtained results were compared with the Original JAYA and JAYA variants. For comparison, the method in the problem definition of the

CEC 2014 benchmark set was used [68].

for $i = 1 : 1000000$

$x = 0.55 + (\text{double})i;$

$x = x + x; x = x/2; x = x * x;$

$x = \text{sqrt}(x); x = \log(x); x = \exp(x); x = x/(x + 2);$

end

According to this method, first, the T_0 time value is obtained by running the given piece of code. The T_1 time

TABLE 16. CEC 2011 functions used in the experiments and their properties.

| Functions | Dimensions | Problems | Application Domain |
|-----------|------------|---|--------------------|
| f1 | 6 | Parameter Estimation for Frequency-Modulated (FM) Sound Waves | Communication |
| f2 | 30 | Lennard-Jones Potential Problem | Chemistry |
| f4 | 1 | Optimal Control of a Non-Linear Stirred Tank Reactor | Chemistry |
| f10 | 12 | Circular Antenna Array Design Problem | Communication |
| F11.1 | 120 | The ELD Problems | Economics |
| F11.3 | 6 | ELD Instance 1 | Economics |
| F11.5 | 15 | ELD Instance 3 | Economics |
| F11.7 | 140 | ELD Instance 5 | Economics |
| F13 | 22 | Cassini 2: Spacecraft Trajectory Optimization Problem | Astronomy |

TABLE 17. Error values obtained by JAYA variant algorithms in real world problems. (The best results are highlighted in bold.)

| Functions | JAYASIP | | JAYA | | CLJAYA | | LJA | | LWIJAYA | | MJA | |
|-----------|-------------------|-----------|------------------|-----------|------------------|-----------|------------------|-----------|------------|-----------|------------|-----------|
| | mean | std | mean | std | mean | std | mean | std | mean | std | mean | std |
| f1 | 1.731E+01 | 5.257E+00 | 1.613E+01 | 3.761E+00 | 1.312E+01 | 3.329E+00 | 1.566E+01 | 4.459E+00 | 1.987E+01 | 2.366E+00 | 2.018E+01 | 3.669E+00 |
| f2 | -1.522E+01 | 4.192E+00 | -7.058E+00 | 7.541E-01 | -1.379E+00 | 3.857E-01 | -9.850E-01 | 2.415E-01 | -1.483E+00 | 4.074E-01 | -1.943E+00 | 4.322E-01 |
| f4 | 1.400E+01 | 2.812E-01 | 1.928E+01 | 3.066E+00 | 1.674E+01 | 3.516E+00 | 1.849E+01 | 3.398E+00 | 1.402E+01 | 1.959E-01 | 1.480E+01 | 1.016E+00 |
| f10 | -1.255E+01 | 2.264E+00 | -1.239E+01 | 2.677E+00 | -3.774E+00 | 1.711E+00 | -2.068E+00 | 1.530E+00 | -4.014E+00 | 1.645E+00 | -5.841E+00 | 1.010E+00 |
| F11.1 | 5.159E+05 | 1.811E+06 | 6.851E+04 | 2.882E+04 | 5.371E+04 | 1.597E+03 | 1.459E+05 | 2.061E+05 | 9.464E+06 | 1.085E+06 | 9.163E+07 | 4.966E+07 |
| F11.3 | 1.549E+04 | 1.870E+01 | 1.657E+04 | 5.413E+03 | 2.090E+05 | 1.560E+05 | 1.550E+04 | 3.545E+01 | 1.554E+04 | 4.508E+01 | 1.557E+04 | 6.188E+01 |
| F11.5 | 3.300E+04 | 1.013E+02 | 3.288E+04 | 1.036E+02 | 4.181E+06 | 3.486E+06 | 3.282E+04 | 7.866E+01 | 4.133E+04 | 1.874E+04 | 3.306E+04 | 1.337E+02 |
| F11.7 | 2.686E+06 | 8.616E+05 | 1.992E+06 | 3.732E+04 | 1.896E+10 | 3.487E+09 | 6.206E+07 | 1.376E+08 | 1.474E+10 | 3.605E+09 | 8.834E+07 | 1.276E+08 |
| F13 | 4.723E+01 | 9.728E+00 | 2.471E+01 | 5.393E+00 | 8.889E+01 | 1.322E+01 | 2.483E+01 | 5.281E+00 | 3.426E+01 | 4.645E+00 | 3.517E+01 | 6.469E+00 |
| rank | 2.44 | | 2.67 | | 4.44 | | 3.33 | | 4.00 | | 4.11 | |

TABLE 18. CEC 2014 computational complexity results for 30 dimensions.

| | T0 | T1 | $\hat{T}2$ | $(\hat{T}2 - T1)/T0$ |
|----------|--------|-------|---------------|----------------------|
| JAYA-SIP | 0.0222 | 0.213 | 0.4294 | 9.7853 |
| MJA | 0.0222 | 0.213 | 1.6303 | 63.9680 |
| LWIJAYA | 0.0222 | 0.213 | 0.7586 | 24.6397 |
| LJA | 0.0222 | 0.213 | 3.8435 | 163.8212 |
| JAYA | 0.0222 | 0.213 | 0.8152 | 27.1902 |
| CLJAYA | 0.0222 | 0.213 | 0.7877 | 25.9490 |

TABLE 19. CEC 2014 computational complexity results for 50 dimensions.

| | T0 | T1 | $\hat{T}2$ | $(\hat{T}2 - T1)/T0$ |
|----------|--------|--------|---------------|----------------------|
| JAYA-SIP | 0.0222 | 0.4401 | 0.7373 | 13.4093 |
| MJA | 0.0222 | 0.4401 | 2.5913 | 97.0620 |
| LWIJAYA | 0.0222 | 0.4401 | 1.2638 | 37.1665 |
| LJA | 0.0222 | 0.4401 | 6.0783 | 254.3885 |
| JAYA | 0.0222 | 0.4401 | 1.3476 | 40.9446 |
| CLJAYA | 0.0222 | 0.4401 | 1.3627 | 41.6300 |

value is then calculated using 200000 function calls of the 18th test function from the CEC 2014 benchmark set. The algorithms then solve the 18th function 200000 times to determine the $T2$ time value. Finally, the $\hat{T}2$ variable is determined by taking the average value of running the algorithms five times. The values obtained as a result of these processes are provided in Table 18 for 30 dimensions and Table 19 for 50 dimensions.

The small $\hat{T}2$ value in Tables 18 and 19 indicates that the algorithm needs less computational time. In addition, $(\hat{T}2 - T1)/T0$ ratio is calculated in these tables. This ratio is also used to express the complexity of algorithms. Here, the smaller value indicates that the algorithm is better. The JAYA-SIP algorithm has less computational complexity than the algorithms with which it is compared, with 9.7853 and 13.4093.

VI. CONCLUSION

In this study, three improvements were proposed for the Original JAYA algorithm. The algorithm was enhanced with senior learning, incremental population strategy, and Powell's method, resulting in a powerful JAYA variant (JAYA-SIP). The performance of the proposed JAYA variant was tested with the CEC 2014 benchmark set for low dimension, the SOCO for large scale and nine CEC 2011 problems for real world problems. The results of the algorithm were compared with JAYA variants and various recent meta-heuristic algorithms. According to the results of the experiment, better results were obtained than the algorithms in which the JAYA-SIP algorithm was compared according to the objective function value.

A future study consideration is the proposed algorithm to binary optimization problems and expensive problems.

REFERENCES

- [1] C. Huang, Y. Li, and X. Yao, "A survey of automatic parameter tuning methods for metaheuristics," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 201–216, Apr. 2020, doi: [10.1109/TEVC.2019.2921598](https://doi.org/10.1109/TEVC.2019.2921598).
- [2] Z. Li, X. Lin, Q. Zhang, and H. Liu, "Evolution strategies for continuous optimization: A survey of the state-of-the-art," *Swarm Evol. Comput.*, vol. 56, Aug. 2020, Art. no. 100694, doi: [10.1016/j.swevo.2020.100694](https://doi.org/10.1016/j.swevo.2020.100694).
- [3] M. Gunduz and M. Aslan, "DJAYA: A discrete Jaya algorithm for solving traveling salesman problem," *Appl. Soft Comput.*, vol. 105, Jul. 2021, Art. no. 107275.
- [4] M. N. Omidvar, X. Li, and K. Tang, "Designing benchmark problems for large-scale continuous optimization," *Inf. Sci.*, vol. 316, pp. 419–436, Sep. 2015, doi: [10.1016/j.ins.2014.12.062](https://doi.org/10.1016/j.ins.2014.12.062).
- [5] G. Wu, X. Wen, L. Wang, W. Pedrycz, and P. N. Suganthan, "A voting-mechanism-based ensemble framework for constraint handling techniques," *IEEE Trans. Evol. Comput.*, vol. 26, no. 4, pp. 646–660, Aug. 2022.
- [6] P. N. Suganthan and R. Katuwal, "On the origins of randomization-based feedforward neural networks," *Appl. Soft Comput.*, vol. 105, Jul. 2021, Art. no. 107239.

- [7] R. S. Al-Gharaibeh, M. Z. Ali, M. I. Daoud, R. Alazrai, H. Abdel-Nabi, S. Hriez, and P. N. Suganthan, "Real-parameter constrained optimization using enhanced quality-based cultural algorithm with novel influence and selection schemes," *Inf. Sci.*, vol. 576, pp. 242–273, Oct. 2021.
- [8] M. Z. Ali, P. N. Suganthan, R. G. Reynolds, and A. F. Al-Badameh, "Leveraged neighborhood restructuring in cultural algorithms for solving real-world numerical optimization problems," *IEEE Trans. Evol. Comput.*, vol. 20, no. 2, pp. 218–231, Apr. 2016.
- [9] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Hum. Sci. (MHS)*, Oct. 1995, pp. 39–43.
- [10] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.
- [11] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, Apr. 2007.
- [12] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *J. Inf. Sci.*, vol. 179, no. 13, pp. 2232–2248, 2009.
- [13] M. S. Kiran, "TSA: Tree-seed algorithm for continuous optimization," *Expert Syst. Appl.*, vol. 42, no. 19, pp. 6686–6698, 2015.
- [14] X-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *Int. J. Bio-Inspired Comput.*, vol. 2, no. 2, pp. 78–84, 2010.
- [15] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.
- [16] G. Yavuz, "Diversified position update equation-based SSA with refreshing-gap strategy for global optimization," *J. Comput. Sci.*, vol. 60, Apr. 2022, Art. no. 101597.
- [17] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.
- [18] N. H. Awad, M. Z. Ali, P. N. Suganthan, and R. G. Reynolds, "CADE: A hybridization of cultural algorithm and differential evolution for numerical optimization," *Inf. Sci.*, vol. 378, pp. 215–241, Feb. 2017.
- [19] G. Yavuz, "L-shade algoritmasının otomatik parametre yapılandırma yöntemi ile iyileştirilmesi," *Bilişim Teknolojileri Dergisi*, vol. 15, no. 2, pp. 189–197, Apr. 2022, doi: [10.17671/gazibtd.1034921](https://doi.org/10.17671/gazibtd.1034921).
- [20] G. Yavuz, "100 basamak probleminin JADE algoritması ile Çözülmesi," *Avrupa Bilim ve Teknoloji Dergisi*, vol. 493, no. 21, pp. 493–500, 2021, doi: [10.31590/ejosat.839083](https://doi.org/10.31590/ejosat.839083).
- [21] R. V. Rao and A. Saroj, "A self-adaptive multi-population based Jaya algorithm for engineering optimization," *Swarm Evol. Comput.*, vol. 37, pp. 1–26, Dec. 2017, doi: [10.1016/j.swevo.2017.04.008](https://doi.org/10.1016/j.swevo.2017.04.008).
- [22] R. V. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *Int. J. Ind. Eng. Comput.*, vol. 7, no. 1, pp. 19–34, 2016.
- [23] W. F. Gao, S. Y. Liu, and L. L. Huang, "A novel artificial bee colony algorithm with Powell's method," *Appl. Soft Comput.*, vol. 13, no. 9, pp. 3763–3775, Sep. 2013.
- [24] S. Zhang and Y. Zhou, "Grey wolf optimizer based on Powell local optimization method for clustering analysis," *Discrete Dyn. Nature Soc.*, vol. 2015, pp. 1–17, Nov. 2015.
- [25] M. Rajeswari, K. Thirugnanasambandam, R. S. Raghav, U. Prabu, D. Saravanan, and D. K. Anguraj, "Flower pollination algorithm with Powell's method for the minimum energy broadcast problem in wireless sensor network," *Wireless Pers. Commun.*, vol. 119, no. 2, pp. 1–25, Feb. 2021.
- [26] M. A. Oca, "Incremental social learning in swarm intelligence algorithms for continuous optimization," in *Computational Intelligence*. Berlin, Germany: Springer, 2013, pp. 31–45.
- [27] M. A. M. de Oca, T. Stutzle, K. Van den Eenden, and M. Dorigo, "Incremental social learning in particle swarms," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 368–384, Apr. 2011.
- [28] T. Liao, M. A. M. D. Oca, D. Aydin, T. Stutzle, and M. Dorigo, "An incremental ant colony algorithm with local search for continuous optimization," in *Proc. 13th Annu. Conf. Genet. Evol. Comput.*, 2011, pp. 125–132.
- [29] S. Özyön, C. Yaşar, and H. Temurtaş, "Incremental gravitational search algorithm for high-dimensional benchmark functions," *Neural Comput. Appl.*, vol. 31, no. 8, pp. 3779–3803, 2019.
- [30] S. Özyön and D. Aydin, "Incremental artificial bee colony with local search to economic dispatch problem with ramp rate limits and prohibited operating zones," *Energ. Convers. Manage.*, vol. 65, pp. 397–407, May 2013.
- [31] D. Aydin, G. Yavuz, and T. Stutzle, "ABC-X: A generalized, automatically configurable artificial bee colony framework," *Swarm Intell.*, vol. 11, no. 1, pp. 1–38, Mar. 2017.
- [32] G. Yavuz, D. Aydin, and T. Stutzle, "Self-adaptive search equation-based artificial bee colony algorithm on the CEC 2014 benchmark functions," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 1173–1180.
- [33] G. Yavuz and D. Aydin, "Improved self-adaptive search equation-based artificial bee colony algorithm with competitive local search strategy," *Swarm Evol. Comput.*, vol. 51, Dec. 2019, Art. no. 100582.
- [34] K. Yu, J. J. Liang, B. Y. Qu, X. Chen, and H. Wang, "Parameters identification of photovoltaic models using an improved JAYA optimization algorithm," *Energ. Convers. Manage.*, vol. 150, pp. 742–753, Oct. 2017, doi: [10.1016/j.enconman.2017.08.063](https://doi.org/10.1016/j.enconman.2017.08.063).
- [35] R. V. Rao and D. P. Rai, "Optimisation of welding processes using quasi-oppositional-based Jaya algorithm," *J. Exp. Theor. Artif. Intell.*, vol. 29, no. 5, pp. 1099–1117, Sep. 2017, doi: [10.1080/0952813X.2017.1309692](https://doi.org/10.1080/0952813X.2017.1309692).
- [36] L. Wang and C. Huang, "A novel elite opposition-based Jaya algorithm for parameter estimation of photovoltaic cell models," *Optik*, vol. 155, pp. 351–356, Feb. 2018, doi: [10.1016/j.ijleo.2017.10.081](https://doi.org/10.1016/j.ijleo.2017.10.081).
- [37] X. Yang and W. Gong, "Opposition-based Jaya with population reduction for parameter estimation of photovoltaic solar cells and modules," *Appl. Soft Comput.*, vol. 104, Jun. 2021, Art. no. 107218.
- [38] N. A. Alawad and B. H. Abed-alguni, "Discrete Jaya with refraction learning and three mutation methods for the permutation flow shop scheduling problem," *J. Supercomput.*, vol. 78, no. 3, pp. 3517–3538, 2022.
- [39] K. K. Ingle and D. R. K. Jatoh, "An efficient Jaya algorithm with Lévy flight for non-linear channel equalization," *Expert Syst. Appl.*, vol. 145, May 2020, Art. no. 112970.
- [40] Z. H. Leghari, M. Y. Hassan, D. M. Said, T. A. Jumani, and Z. A. Memon, "A novel grid-oriented dynamic weight parameter based improved variant of Jaya algorithm," *Adv. Eng. Softw.*, vol. 150, Dec. 2020, Art. no. 102904, doi: [10.1016/j.advengsoft.2020.102904](https://doi.org/10.1016/j.advengsoft.2020.102904).
- [41] T. V. Luu and N. S. Nguyen, "Parameters extraction of solar cells using modified Jaya algorithm," *Optik*, vol. 203, Feb. 2020, Art. no. 164034.
- [42] X. Jian and Z. Weng, "A logistic chaotic Jaya algorithm for parameters identification of photovoltaic cell and module models," *Optik*, vol. 203, Feb. 2020, Art. no. 164041, doi: [10.1016/j.ijleo.2019.164041](https://doi.org/10.1016/j.ijleo.2019.164041).
- [43] R. V. Rao and H. S. Keesari, "Multi-team perturbation guiding Jaya algorithm for optimization of wind farm layout," *Appl. Soft Comput.*, vol. 71, pp. 800–815, Oct. 2018.
- [44] A. Farah and A. Belazi, "A novel chaotic Jaya algorithm for unconstrained numerical optimization," *Nonlinear Dyn.*, vol. 93, no. 3, pp. 1451–1480, 2018.
- [45] K. Gholami, H. Olfat, and J. Gholami, "An intelligent hybrid Jaya and crow search algorithms for optimizing constrained and unconstrained problems," *Soft Comput.*, vol. 25, no. 22, pp. 14393–14411, Nov. 2021.
- [46] S. S. Alotaibi, "Optimization insisted watermarking model: Hybrid firefly and Jaya algorithm for video copyright protection," *Soft Comput.*, vol. 24, no. 19, pp. 14809–14823, Mar. 2020, doi: [10.1007/s00500-020-04833-8](https://doi.org/10.1007/s00500-020-04833-8).
- [47] S. K. Goudos, T. V. Yioultis, A. D. Boursianis, K. E. Psannis, and K. Siakavara, "Application of new hybrid Jaya grey wolf optimizer to antenna design for 5G communications systems," *IEEE Access*, vol. 7, pp. 71061–71071, 2019, doi: [10.1109/ACCESS.2019.2919116](https://doi.org/10.1109/ACCESS.2019.2919116).
- [48] G. Xiong, J. Zhang, D. Shi, L. Zhu, and X. Yuan, "Optimal identification of solid oxide fuel cell parameters using a competitive hybrid differential evolution and Jaya algorithm," *Int. J. Hydrogen Energy*, vol. 46, no. 9, pp. 6720–6733, Feb. 2021, doi: [10.1016/j.ijhydene.2020.11.119](https://doi.org/10.1016/j.ijhydene.2020.11.119).
- [49] A. Kaur, S. Sharma, and A. Mishra, "A novel Jaya-BAT algorithm based power consumption minimization in cognitive radio network," *Wireless Pers. Commun.*, vol. 108, no. 4, pp. 2059–2075, Oct. 2019, doi: [10.1007/s11277-019-06509-5](https://doi.org/10.1007/s11277-019-06509-5).
- [50] V. Kumar and S. M. Yadav, "Optimization of reservoir operation with a new approach in evolutionary computation using TLBO algorithm and Jaya algorithm," *Water Resour. Manage.*, vol. 32, no. 13, pp. 4375–4391, Oct. 2018.
- [51] M. Azizi, S. A. M. Ghasemi, R. G. Ejlali, and S. Talatahari, "Optimum design of fuzzy controller using hybrid ant lion optimizer and Jaya algorithm," *Artif. Intell. Rev.*, vol. 53, no. 3, pp. 1553–1584, Mar. 2020, doi: [10.1007/s10462-019-09713-8](https://doi.org/10.1007/s10462-019-09713-8).
- [52] M. F. Tefek and M. Beşkirli, "JayaL: A novel Jaya algorithm based on elite local search for optimization problems," *Arabian J. Sci. Eng.*, vol. 46, no. 9, pp. 8925–8952, Sep. 2021, doi: [10.1007/s13369-021-05677-6](https://doi.org/10.1007/s13369-021-05677-6).

- [53] S. Gupta, N. Kumar, and L. Srivastava, "An efficient Jaya algorithm with Powell's pattern search for optimal power flow incorporating distributed generation," *Energy Sources B, Econ. Plan. Policy*, vol. 16, no. 8, pp. 759–786, 2021.
- [54] M. Aslan, M. Gunduz, and M. S. Kiran, "JayaX: Jaya algorithm with XOR operator for binary optimization," *Appl. Soft Comput.*, vol. 82, Sep. 2019, Art. no. 105576.
- [55] E. H. Houssein, A. G. Gad, and Y. M. Wazery, "Jaya algorithm and applications: A comprehensive review," *Metaheuristics and Optimization in Computer and Electrical Engineering*. Cham, Switzerland: Springer, 2021, pp. 3–24.
- [56] R. V. Rao, *Jaya: An Advanced Optimization Algorithm and Its Engineering Applications*. Cham, Switzerland: Springer, 2019, pp. 770–780.
- [57] R. A. Zitar, M. A. Al-Betar, M. A. Awadallah, I. A. Doush, and K. Assaleh, "An intensive and comprehensive overview of JAYA algorithm, its Versions and Applications," *Arch. Comput. Methods Eng.*, vol. 29, pp. 763–792, Mar. 2022.
- [58] M. Premkumar, P. Jangir, R. Sowmya, R. M. Elavarasan, and B. S. Kumar, "Enhanced chaotic Jaya algorithm for parameter estimation of photovoltaic cell/modules," *ISA Trans.*, vol. 116, pp. 139–166, Oct. 2021.
- [59] M. Liu, Z. Cao, J. Zhang, L. Wang, C. Huang, and X. Luo, "Short-term wind speed forecasting based on the Jaya-SVM model," *Int. J. Electr. Power Energy Syst.*, vol. 121, Oct. 2020, Art. no. 106056.
- [60] W. Warid, "Optimal power flow using the AMTPG-Jaya algorithm," *Appl. Soft Comput.*, vol. 91, Jun. 2020, Art. no. 106252.
- [61] S. O. Degertekin, L. Lamberti, and I. B. Ugur, "Discrete sizing/layout/topology optimization of truss structures with an advanced Jaya algorithm," *Appl. Soft Comput.*, vol. 79, pp. 363–390, Jun. 2019.
- [62] R. V. Rao and A. Saroj, "Constrained economic optimization of shell-and-tube heat exchangers using elitist-Jaya algorithm," *Energy*, vol. 128, pp. 785–800, Jun. 2017.
- [63] K. Thirumoorthy and K. Muneeswaran, "A hybrid approach for text document clustering using Jaya optimization algorithm," *Expert Syst. Appl.*, vol. 178, Sep. 2021, Art. no. 115040.
- [64] A. Chaudhuri and T. P. Sahu, "A hybrid feature selection method based on binary Jaya algorithm for micro-array data classification," *Comput. Electr. Eng.*, vol. 90, Mar. 2021, Art. no. 106963.
- [65] G. Iacca, V. C. D. S. Junior, and V. V. de Melo, "An improved Jaya optimization algorithm with Lévy flight," *Expert Syst. Appl.*, vol. 165, Mar. 2021, Art. no. 113902.
- [66] R. P. Brent, *Algorithms for Minimization Without Derivatives*. Chelmsford, MA, USA: Courier Corporation, 2013.
- [67] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *Comput. J.*, vol. 7, no. 2, pp. 155–162, Jan. 1964.
- [68] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," *Comput. Intell. Lab., Nanyang Technol. Univ., Zhengzhou Univ., Zhengzhou, China, Tech. Rep.*, 2013, p. 490, vol. 635.
- [69] M. Lozano, D. Molina, and F. Herrera, "Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems," *Soft Comput.*, vol. 15, no. 11, pp. 2085–2087, Nov. 2011.
- [70] S. Das and P. N. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems," *School Elect. Electron. Eng., Dept. Electron. Telecommun. Eng., Jadavpur Univ., Nanyang Technol. Univ., Kolkata, India, Tech. Rep.*, 2010, pp. 341–359.
- [71] Y. Zhang, M. Ma, and Z. Jin, "Comprehensive learning Jaya algorithm for parameter extraction of photovoltaic models," *Energy*, vol. 211, Nov. 2020, Art. no. 118644, doi: [10.1016/j.energy.2020.118644](https://doi.org/10.1016/j.energy.2020.118644).
- [72] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014, doi: [10.1016/j.advengsoft.2013.12.007](https://doi.org/10.1016/j.advengsoft.2013.12.007).
- [73] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 2016, doi: [10.1016/j.knsys.2015.12.022](https://doi.org/10.1016/j.knsys.2015.12.022).
- [74] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, L. P. Cáceres, and M. Birattari, "The irace package: Iterated racing for automatic algorithm configuration," *Oper. Res. Perspect.*, vol. 3, pp. 43–58, Sep. 2016.
- [75] R. Poláková, J. Tvrđík, and P. Bujok, "Controlled restart in differential evolution applied to CEC2014 benchmark functions," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 2230–2236.



GÜRCAN YAVUZ received the B.Sc. degree from the Department of Computer Engineering and the M.Sc. degree from the Department of Electrical and Electronic Engineering, Kütahya Dumlupınar University, in 2009 and 2013, respectively, and the Ph.D. degree from Eskişehir Technical University, Turkey, in July 2019. He is currently an Assistant Professor. His research interests include artificial intelligence, swarm intelligence, and metaheuristics.

• • •