Journal of Intelligent Manufacturing (2022) 33:1229–1253
https://doi.org/10.1007/s10845-020-01723-6

Comprehensive learning Jaya algorithm for engineering design optimization problems

Yiying Zhang[1] · Zhigang Jin[1]

Received: 22 March 2020 / Accepted: 3 December 2020 / Published online: 3 January 2021

Abstract
Jaya algorithm (JAYA) is a recently developed metaheuristic algorithm for global optimization problems. JAYA has a very simple structure and only needs the essential population size and terminal condition for solving optimization problems. However, JAYA is easy to get trapped in the local optimum for solving complex global optimization problems due to its single learning strategy. Motivated by this disadvantage of JAYA, this paper presents an improved JAYA, named comprehensive learning JAYA algorithm (CLJAYA), for solving engineering design optimization problems. The core idea of CLJAYA is the designed comprehensive learning mechanism by making full use of population information. The designed comprehensive learning mechanism consists of three different learning strategies to improve the global search ability of JAYA. To investigate the performance of CLJAYA, CLJAYA is first evaluated by the well-known CEC 2013 and CEC 2014 test suites, which include 50 multimodal test functions and eight unimodal test functions. Then CLJAYA is employed to solve five real-world engineering optimization problems. Experimental results demonstrate that CLJAYA can achieve better solutions for most test problems than JAYA and the other compared algorithms, which indicates the designed comprehensive learning mechanism is very effective. In addition, the source code of the proposed CLJAYA can be loaded from https://www.mathworks.com/matlabcentral/fileexchange/82134-the-source-code-for-cljaya.

Keywords Jaya algorithm · Comprehensive learning · Metaheuristic algorithm · Engineering optimization

List of symbols

$\mathbf{X}$	Population
N	Population size
D	Dimension
$\mathbf{x}_i$	The position of the ith individual
$\mathbf{v}_i$	The candidate position of the ith individual
$\mathbf{x}_{\mathrm{BEST},j}$	The jth variable of the best individual
$\mathbf{x}_{\mathrm{WORST},j}$	The jth variable of the worst individual
$\mathbf{M}$	The mean position of the population
p_{switch}	Switch probability
p, q	Integers between 1 and N
$\kappa_1, \kappa_2, \varphi_5, \varphi_6, \chi$	Random numbers between 0 and 1 with uniform distribution
$\varphi_1, \varphi_2, \varphi_3, \varphi_4$	Random numbers with standard normal distribution
$\mathbf{L}$	The lower boundary of the variables
$\mathbf{U}$	The upper boundary of the variables
F_{current}	The current number of function evaluations
$F_{\max}$	The maximum number of function evaluations
R_N	The number of independent runs
$\mathbf{x}^*$	The real optimal solution
h_t	The tth equality constraint
g_k	The kth inequality constraint
m	The number of equality constraints
n	The number of inequality constraints
$P(\mathbf{x})$	The total penalty function
$H_i(\mathbf{x})$	The penalty function of the ith equality constraint
$G_j(\mathbf{x})$	The penalty function of the jth inequality constraint
η_i	The penalty factor of the penalty function of the ith equality constraint

✉ Zhigang Jin
zgjin@tju.edu.cn

Yiying Zhang
zhangyiying@tju.edu.cn

[1] School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, People's Republic of China

ξ_j	The penalty factor of the penalty function of the *j*th inequality constraint
MEAN	Mean absolute error
STD	Standard variance
NNA	Neural network algorithm
GWO	Grey wolf optimizer
WOA	Whale optimization algorithm
SCA	Sine cosine algorithm
JAYA	Jaya algorithm
TLBO	Teaching–learning-based optimization
CLJAYA	Comprehensive learning Jaya algorithm

Introduction

Engineering optimization is an attractive and challenging field of study. An engineering design problem usually includes the following components: objective function, design variables, feasible solutions and constrained conditions. The feasible solutions are the set of all possible values of the design variables. Solving an engineering optimization problem is to find the best solution meeting the constrained conditions from a large of feasible solutions by an optimization technique. Various numerical optimization methods have been proposed to solve engineering optimization problems. Numerical methods usually require substantial gradient information and are sensitive to the initial solutions (Cheng and Prayogo 2014; Eskandar et al. 2012; Liu et al. 2019). In fact, most real-world engineering optimization problems are very complex, whose objective functions usually have more than one local optimum. Gradient search in these problems is difficult and unstable(Cheng and Prayogo 2014; Lee and Geem 2004). Thus these numerical methods may easily get trapped in the local optima for complex engineering optimization problems(Eskandar et al. 2012). Given the drawbacks of the numerical methods, it is necessary for researchers to design simple and efficient optimization methods for real-world engineering optimization problems. Metaheuristic algorithms are developed under this background.

Briefly, metaheuristic methods commonly operate by combing some defined rules and randomness to simulate natural phenomena (Lee and Geem 2005). From the inspiration source, the reported metaheuristic algorithms can be broadly classified into the following four categories:

- Evolutionary algorithms. The inspiration source of these algorithms is biological evolution. Differential evolution (Storn and Price 1997) and genetic algorithm (Holland 1975) are two typical members of such algorithms. Differential evolution and genetic algorithm perform the search tasks by simulating some processes of biological evolution.
- Swarm intelligence algorithms. These algorithms mimic some behavior of animals and plants in nature, such as foraging behavior in particle swarm optimization (Kennedy and Eberhart 1995) and hunting mechanism of grey wolves in grey wolf optimizer (Mirjalili et al. 2014).
- Physics-based algorithms. These algorithms are inspired from some physical phenomenon in real life, such as the law of gravity in gravitational search algorithm (Rashedi et al. 2009) and the water cycle process and how rivers and streams flow to the sea in water cycle algorithm (Eskandar et al. 2012).
- Human-related algorithms. These algorithms are inspired from human activity, such as the artificial nervous networks in neural network algorithm (Sadollah et al. 2018) and the teaching activities in teaching–learning-based optimization (Rao et al. 2011).

Although many metaheuristic algorithms have been successfully applied to solve a lot of real-world engineering optimization problems, there remains a need for developing simple and efficient metaheuristic algorithms without any effort for fine tuning initial parameters due to the following two reasons:

- Most reported metaheuristic algorithms all need special parameters. The parameters of metaheuristic algorithms consist of common parameters and special parameters. Every metaheuristic algorithm needs common parameters, such as population size and stopping criterion (e.g. the maximum number of function evaluations, the maximum number of iterations or the defined threshold value). The parameters reflecting the characteristics of algorithms can be called special parameters, such as differential amplification factor and crossover probability in differential evolution (Storn and Price 1997), discovery probability in cuckoo search (Yang and Deb 2014), and cognitive factor and social factor in particle swarm optimization (Kennedy and Eberhart 1995). The major drawbacks of metaheuristic algorithms with special parameters can be summarized as follows: (1) it is very hard task to set the optimal values of these parameters for unknown optimization problems; (2) different optimization problems usually need different optimal values for these parameters to get the optimal solutions. Given the two drawbacks, the applications of metaheuristic algorithms with special parameters will be restricted. To overcome the two drawbacks, developing metaheuristic algorithms without special parameters is a very efficient method.
- There is a very important theory in the optimization field, which is called the No Free Lunch (NFL) theorem(Wolpert and Macready 1997). According to NFL theorem, a metaheuristic algorithm may obtain very promising results

on a set of optimization problems while it may show poor performance on another set of optimization problems. In other words, no single metaheuristic algorithm is suitable for solving all optimization problems. Thus, more studies are very necessary for researcher to develop new optimization algorithms for solving different types of real-world engineering optimization problems.

Motivated by the mentioned reasons, this work reports a new metaheuristic method without special parameter, named comprehensive learning Jaya algorithm (CLJAYA), for solving engineering design optimization problems. CLJAYA is an improved version of Jaya algorithm (JAYA)(Rao 2016), which is aiming at enhancing the global search ability of JAYA by the designed comprehensive learning mechanism with three different learning strategies. Learning strategy-I in CLJAYA inherits the feature of JAYA. Learning strategy-II introduces the current mean solution to increase the chance of JAYA to escape from the local optima. Learning strategy-III is guided by the current best solution to accelerate the convergence speed of JAYA. Obviously, compared with JAYA, CLJAYA can use population information more efficiently to generate the next generation population. To sum up, the contributions of this work are presented as follows:

- A novel optimization algorithm called CLJAYA algorithm is proposed.
- A comprehensive learning mechanism consisting of three different learning strategies is built.
- CLJAYA is evaluated by the well-known CEC 2013 and CEC 2014 test suites.
- CLJAYA is employed to solve five real-world engineering design optimization problems.

The rest of this paper is organized as follows: Sect. 2 presents the brief introduction of JAYA. CLJAYA is described in Sect. 3. CLJAYA is checked by CEC 2013 and CEC 2014 test suites in Sect. 4. CLJAYA is used for solving five real-world engineering optimization problems in Sect. 5. Finally, conclusions and further work are made in Sect. 6.

Jaya algorithm

JAYA has a very simple learning strategy to perform the search process, which can be stated as follows. Let $\mathbf{X}$ is a population consisting of N individuals, i.e. $\mathbf{X}=\left[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_N\right]$. Assume there are D considered variables for the given problem, i.e. $\mathbf{x}_i=\left\{\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \mathbf{x}_{i,3}, \ldots, \mathbf{x}_{i,D}\right\}$, $i = 1, 2, 3, \ldots, N$. In JAYA, the position of the ith individual can be updated by

$$\begin{aligned}\mathbf{v}_{i,j} = \mathbf{x}_{i,j} + \kappa_1 \times (\mathbf{x}_{\mathrm{BEST},j} - \left|\mathbf{x}_{i,j}\right|) \\ - \kappa_2 \times (\mathbf{x}_{\mathrm{WORST},j} - \left|\mathbf{x}_{i,j}\right|), i = 1, 2, 3, \ldots, N, \\ j = 1, 2, 3, \ldots, D \end{aligned} \tag{1}$$

where κ_1 and κ_2 are two random numbers between 0 and 1 subject to uniform distribution, $\mathbf{v}_i$ is the candidate position of the ith individual, $\mathbf{x}_{\mathrm{BEST},j}$ is the value of the jth variable in the current best individual, and $\mathbf{x}_{\mathrm{WORST},j}$ is the value of the jth variable in the current worst individual. According to the authors of JAYA(Rao 2016), the second and third terms on right-hand side of Eq. (1) indicate the tendency of the solution $\mathbf{x}_i$ to move closer to the best solution and avoid the worst solution, respectively. To find the optimal solution with a fast speed, the final position of the ith individual at this iteration is selected from the candidate position $\mathbf{v}_i$ and $\mathbf{x}_i$, which can be expressed as

$$\mathbf{x}_i=\begin{cases} \mathbf{v}_i, \text{if } f(\mathbf{v}_i) < f(\mathbf{x}_i) \\ \mathbf{x}_i, \text{otherwise} \end{cases} \tag{2}$$

The proposed CLJAYA

This section presents the proposed CLJAYA in detail. The framework of CLJAYA is shown in Fig. 1. As can be seen from Fig. 1, updating population in CLJAYA is completed by the designed comprehensive learning mechanism with three different learning strategies. Thus, we first introduce the motivation of the designed comprehensive learning mechanism in "Motivation of CLJAYA" section. Then, the comprehensive learning of CLJAYA is given in The implementation of CLJAYAThe implementation of CLJAYA" section.

Motivation of CLJAYA

JAYA has two drawbacks that may result in its weak ability of avoiding the local optimum, which can be summarized as follows in detail:

- JAYA doesn't make full use of population information. As shown in Eq. (1), JAYA has only one learning strategy, which employs the current best solution and the current worst solution to guide the search direction of the population. Thus once the current best individual is trapped into a local optimum, the other individuals will be attracted to approach this local optimum gradually based on Eq. (1). This case will cause the loss of

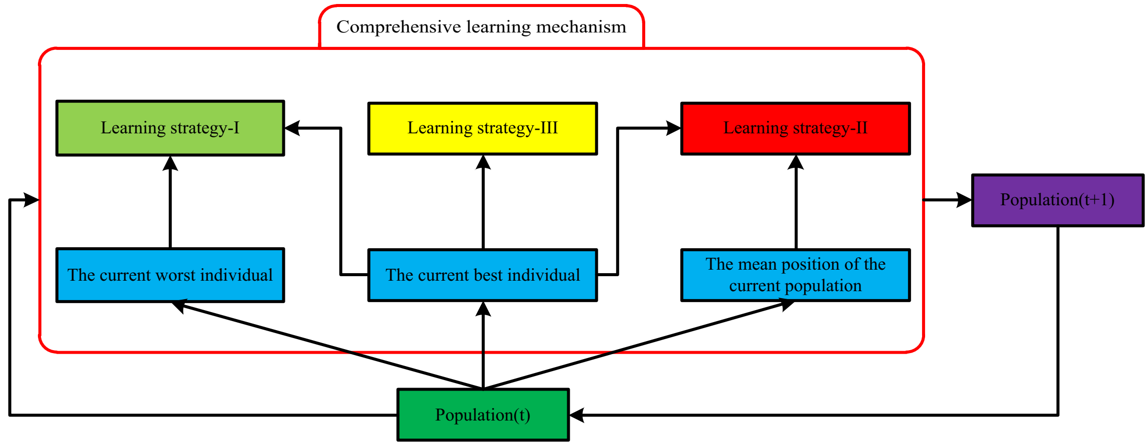

Fig. 1 The framework of the proposed CLJAYA

population diversity. Therefore, it is very difficult for the population to escape from the local optimum.

- The effectiveness of the search operator in JAYA may be tempered in solving optimization problems with search space with positive numbers. In Eq. (1), the absolute value symbol is very critical in keeping population diversity. Generally, the values of the design variables of the real-world engineering optimization problems are more than 0, which means the absolute value symbol is invalid for solving these problems. That is, Eq. (1) can be rewritten as

$$\mathbf{v}_{i,j} = \mathbf{x}_{i,j} + \kappa_1 \times (\mathbf{x}_{\mathrm{BEST},j} - \mathbf{x}_{i,j}) - \kappa_2 \times (\mathbf{x}_{\mathrm{WORST},j} - \mathbf{x}_{i,j}), i = 1, 2, 3, \ldots, N, \quad j = 1, 2, 3, \ldots, D \tag{3}$$

Note that there are the following two risks in Eq. (3): (1) if the ith individual is equal to the current optimal individual, the second term on right-hand side of Eq. (3) is 0, which is of no help to search better solution; (2) if the ith individual is equal to the current worst individual, the third term on right-hand side of Eq. (3) is 0, which also does nothing to find better solution. Obviously, when the mentioned two cases happen, the search ability of JAYA will be reduced.

The above mentioned two disadvantages of JAYA motivate us to design an improved version of JAYA with better global search ability.

The implementation of CLJAYA

Given the disadvantages of the learning strategy in JAYA, a comprehensive learning mechanism consisting of three different learning strategies is built to improve the global search ability of JAYA. The three different learning strategies can be described as:

- Learning strategy-I. This learning strategy is based on the current optimal individual and the current worst individual, which inherits the feature of JAYA and can be denoted as

$$\mathbf{v}_{i,j}=\mathbf{x}_{i,j} + \varphi_1 \times (\mathbf{x}_{\mathrm{BEST},j} - \left|\mathbf{x}_{i,j}\right|) - \varphi_2 \times (\mathbf{x}_{\mathrm{WORST},j} - \left|\mathbf{x}_{i,j}\right|) \tag{4}$$

where φ_1 and φ_2 are two random numbers subject to standard normal distribution. Here, it should be pointed out that Eq. (4) uses two random number (i.e. φ_1 and φ_2) with standard normal distribution while Eq. (1) employs two random numbers (i.e. κ_1 and κ_2) with uniform distribution. Compared with random numbers with uniform distribution, random numbers with standard normal distribution have the larger amplitude of variation, which can extend the search space of the individual. Thus, Eq. (4) has more chance to find better solutions than Eq. (1).

- Learning strategy-II. As an effective indicator of evaluating population distribution, the mean position of the current population has been employed by many optimization algorithms (Cheng and Jin 2015; Rao et al. 2012) to improve their search ability due to the following reason. With the increasing of iteration times, most individuals have gathered around the current optimal individual to perform the local exploitation. The rest few individuals (lagged individuals) are away from the current optimal individual, which perform the task of global exploration. In the search process, the mean position of the current population is always moving. Thus once the population is trapped into local optimum, the lagged individuals guided by the mean position of the current population can have more chance to escape from the local optimum. Given this, the learning strategy-II is designed based on the current optimal individual and the mean position of the current population, which can be defined as

$$\mathbf{v}_{i,j} = \mathbf{x}_{i,j} + \varphi_3 \times (\mathbf{x}_{\mathrm{BEST},j} - |\mathbf{x}_{i,j}|) - \varphi_4 \times (\mathbf{M} - |\mathbf{x}_{i,j}|) \quad (5)$$

where φ_3 and φ_4 are two random numbers subject to standard normal distribution, and $\mathbf{M}$ is the mean position of the current population that can be written as

$$\mathbf{M} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \quad (6)$$

Note that φ_3 and φ_4 in Eq. (5) play the same role with φ_1 and φ_2 in Eq. (4).

- Learning strategy-III. To accelerate the convergence speed, the current optimal individual is considered as a leader in CLJAYA, which can be expressed as

$$\mathbf{v}_{i,j} = \mathbf{x}_{i,j} + \varphi_5 \times (\mathbf{x}_{\mathrm{BEST},j} - \mathbf{x}_{i,j}) + \varphi_6 \times (\mathbf{x}_{p,j} - \mathbf{x}_{q,j}) \quad (7)$$

where φ_5 and φ_6 are two random numbers between 0 and 1 subject to uniform distribution, and p and $q(p \neq q \neq i)$ are two random integers between 1 and N. In addition, considering the case where $\mathbf{x}_i$ is the current best solution, the second term on right-hand side of Eq. (7) is 0. Thus a random perturbed term is added to Eq. (7) to avoid this case.

Learning strategy-I, learning strategy-II, and learning strategy-III have the same importance for CLJAYA and they should be assigned the same selected probability. Given this, the designed comprehensive learning mechanism can be indicated as

$$\mathbf{v}_{i,j} = \begin{cases} \mathbf{x}_{i,j} + \varphi_1 \times (\mathbf{x}_{\mathrm{BEST},j} - |\mathbf{x}_{i,j}|) - \varphi_2 \times (\mathbf{x}_{\mathrm{WORST},j} - |\mathbf{x}_{i,j}|), & \text{if } 0 \le p_{\mathrm{switch}} \le 1/3 \\ \mathbf{x}_{i,j} + \varphi_3 \times (\mathbf{x}_{\mathrm{BEST},j} - |\mathbf{x}_{i,j}|) - \varphi_4 \times (\mathbf{M} - |\mathbf{x}_{i,j}|), & \text{if } 1/3 \le p_{\mathrm{switch}} \le 2/3 \\ \mathbf{x}_{i,j} + \varphi_5 \times (\mathbf{x}_{\mathrm{BEST},j} - \mathbf{x}_{i,j}) + \varphi_6 \times (\mathbf{x}_{p,j} - \mathbf{x}_{q,j}), & \text{if } 2/3 \le p_{\mathrm{switch}} \le 1 \end{cases} \quad (8)$$

where p_{switch} is called switch probability and it is uniformly distributed on the interval from 0 to 1.

The built comprehensive learning mechanism as shown in Eq. (8) is the core idea of CLJAYA. Note that there are no any extra parameters in the built mechanism, which indicates

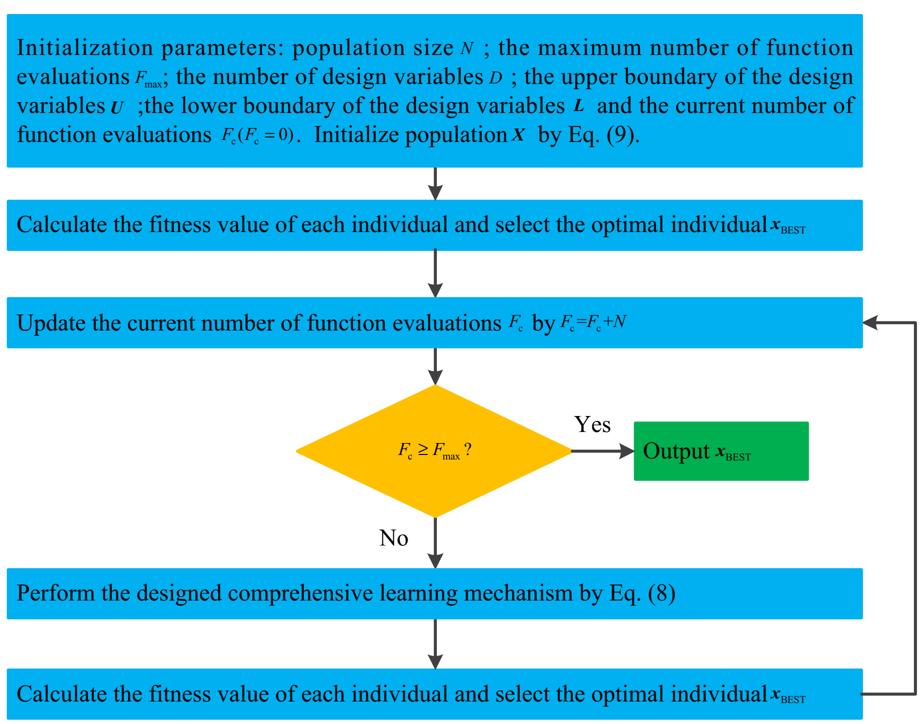

Fig. 2 The flow chart of the proposed CLJAYA

Table 1 The definition of CEC 2013 test suite

No	Type	Name	Dimension	Limits	Optimum
F1	Unimodal functions	Sphere function	30	[– 100,100]	– 1400
F2		Rotated high conditional elliptic function	30	[– 100,100]	– 1300
F3		Rotated bent cigar function	30	[– 100,100]	– 1200
F4		Rotated discus function	30	[– 100,100]	– 1100
F5		Different powers function	30	[– 100,100]	– 1000
F6	Multimodal functions	Rotated rosenbrock's function	30	[– 100,100]	– 900
F7		Rotated schaffers F7 function	30	[– 100,100]	– 800
F8		Rotated ackley's function	30	[– 100,100]	– 700
F9		Rotated weierstrass function	30	[– 100,100]	– 600
F10		Rotated Griewank's function	30	[– 100,100]	– 500
F11		Rastrigin's function	30	[– 100,100]	– 400
F12		Rotated rastrigin's function	30	[– 100,100]	– 300
F13		Non– continuous rotated rastrigin's function	30	[– 100,100]	– 200
F14		Schwefel's function	30	[– 100,100]	– 100
F15		Rotated schwefel's function	30	[– 100,100]	100
F16		Rotated katsuura function	30	[– 100,100]	200
F17		LunacekBi_Rastrigin function	30	[– 100,100]	300
F18		Expanded LunacekBi_Rastrigin function	30	[– 100,100]	400
F19		Expanded griewank's + rosenbrock's function	30	[– 100,100]	500
F20		Expanded scaffer's F6 Function	30	[– 100,100]	600
F21	Composition functions	Composition function 1 (n = 5, rotated)	30	[– 100,100]	700
F22		Composition function 2 (n = 5, rotated)	30	[– 100,100]	800
F23		Composition function 3 (n = 5, rotated)	30	[– 100,100]	900
F24		Composition function 4 (n = 5, rotated)	30	[– 100,100]	1000
F25		Composition function 5 (n = 5, rotated)	30	[– 100,100]	1100
F26		Composition function 6 (n = 5, rotated)	30	[– 100,100]	1200
F27		Composition function 7 (n = 5, rotated)	30	[– 100,100]	1300
F28		Composition function 8 (n = 5, rotated)	30	[– 100,100]	1400

the proposed CLJAYA still inherits the advantages of JAYA, i.e. simple structure and only needs essential parameters. In addition, like JAYA, the population **X** is initialized by

$$\mathbf{x}_{i,j} = l_j + (u_j - l_j) \times \chi,\ i = 1,2,3,\ldots,N, j = 1,2,3,\ldots,D \tag{9}$$

where χ is a random number between 0 and 1 subject to the uniform distribution. Figure 2 shows the flow chart of the proposed CLJAYA.

Applications of CLJAYA on numerical optimization

In this section, the performance of CLJAYA on CEC 2013 and CEC 2014 test suites is checked by comparing with six state-of-the-art metaheuristic algorithms.

As listed in Tables 1–2, the solved test suites have been widely used to test the performance of many metaheuristic algorithms (Li et al. 2015; K.S. and Murugan 2017; Tanweer et al. 2016; Xiang et al. 2019; Zhang et al. 2019). CEC 2013 test suite consists of five unimodal functions (F1–F5), 15 simple multimodal functions (F6–F20) and eight composition functions (F21–F28). CEC2014 test suite includes three unimodal functions (F29–F31), 13 simple multimodal functions (F32–F44) and 14 hybrid functions (F45–F58). Compared with unimodal functions, multimodal functions with more than one local optimal solutions are more complex. Note that composition functions in CEC 2013 test suite and hybrid functions in CEC 2014 test suite are also multimodal functions. That is, 23 of 28 functions in CEC 2013 test suite and 27 of 30 functions in CEC 2014 test suite are multimodal functions. Therefore, the two test suites are very suitable for testing the performance of CLJAYA in solving complex optimization problems. In addition, the detailed information for the two test suites can be found in https://www.ntu.edu.sg/home/EPNSugan/.

CLJAYA is compared with six state-of-the-art metaheuristic algorithms to validate the competitive performance of CLJAYA. The selected algorithms are closely associated with

Table 2 The definition of CEC 2014 test suite

No	Type	Name	Dimension	Limits	Optimum
F29	Unimodal functions	Rotated high conditioned elliptic function	30	[− 100,100]	100
F30		Rotated bent cigar function	30	[− 100,100]	200
F31		Rotated discus function	30	[− 100,100]	300
F32	Multimodal functions	Shifted and rotated Rosenbrocks function	30	[− 100,100]	400
F33		Shifted and rotated Ackleys function	30	[− 100,100]	500
F34		Shifted and rotated Weierstrass function	30	[− 100,100]	600
F35		Shifted and rotated Griewanks function	30	[− 100,100]	700
F36		Shifted Rastrigins function	30	[− 100,100]	800
F37		Six Hump Camel Back	30	[− 100,100]	900
F38		Shifted and rotated Rastrigins function	30	[− 100,100]	1000
F39		Shifted and rotated Schwefels function	30	[− 100,100]	1100
F40		Shifted and rotated Katsuura function	30	[− 100,100]	1200
F41		Shifted and rotated HappyCat function	30	[− 100,100]	1300
F42		Shifted and rotated HGBat function	30	[− 100,100]	1400
F43		Shifted and rotated Expanded Griewanks + Rosenbrocks function	30	[− 100,100]	1500
F44		Shifted and rotated Expanded Scaffers function	30	[− 100,100]	1600
F45	Hybrid functions	Hybrid function 1 (m = 3)	30	[− 100,100]	1700
F46		Hybrid function 2 (m = 3)	30	[− 100,100]	1800
F47		Hybrid function 3 (m = 4)	30	[− 100,100]	1900
F48		Hybrid function 4 (m = 4)	30	[− 100,100]	2000
F49		Hybrid function 5 (m = 5)	30	[− 100,100]	2100
F50		Hybrid function 6 (m = 5)	30	[− 100,100]	2200
F51	Composition functions	Composition function 1 (m = 5)	30	[− 100,100]	2300
F52		Composition function 2 (m = 3)	30	[− 100,100]	2400
F53		Composition function 3 (m = 3)	30	[− 100,100]	2500
F54		Composition function 4 (m = 5)	30	[− 100,100]	2600
F55		Composition function 5 (m = 5)	30	[− 100,100]	2700
F56		Composition function 6 (m = 5)	30	[− 100,100]	2800
F57		Composition function 7 (m = 3)	30	[− 100,100]	2900
F58		Composition function 8 (m = 3)	30	[− 100,100]	3000

CLJAYA in terms of parameters, which include JAYA, teaching–learning-based optimization (TLBO) (Rao et al. 2012), neural network algorithm (NNA) (Sadollah et al. 2018), grey wolf optimizer (GWO)(Mirjalili et al. 2014), whale optimization algorithm (WOA) (Mirjalili and Lewis 2016) and sine cosine algorithm (SCA) (Mirjalili 2016). JAYA is the basis of our proposed CLJAYA. TLBO is a recently proposed metaheuristic algorithm, which is inspired by the traditional teaching method in the classroom. NNA is one of the latest metaheuristic algorithms and its motivation is the artificial neural networks and biological nervous systems. When solving optimization problems, JAYA, NNA and TLBO need the same parameters (i.e. population size and terminal condition) with CLJAYA. GWO, WOA and SCA are inspired by the hunting behavior of grey wolves, the social behavior of humpback whales and the sine cosine theory, respectively. Although the required parameters (i.e. population size and terminal condition) of GWO, WOA and SCA are the same with CLJAYA, there are control parameters related to the terminal condition in the three algorithms. These control parameters can be found in the corresponding references.

In order to make a fair comparison, population size and the maximum number of function evaluations for CLJAYA and the compared algorithms were set to 20 and 300,000, respectively. In addition, every algorithm for every test function was executed 50 independent runs and then the mean absolute error (MEAN) and standard variance (STD) were recorded. The results are presented in Tables 3 and 6. MEAN can be defined by

$$\mathrm{MEAN} = \frac{1}{R_{\mathrm{N}}} \sum_{i=1}^{R_{\mathrm{N}}} \left| f(\mathbf{x}_{\mathrm{Best},i}) - f(\mathbf{x}^{*}) \right| \tag{10}$$

Table 3 The statistical results obtained by CLJAYA and the compared algorithms on CEC 2013 test suite

No	Index	NNA	GWO	WOA	SCA	JAYA	TLBO	CLJAYA
F1	MEAN	1.625E+01	2.011E+03	4.051E−01	1.129E+04	5.039E+03	1.231E−06	6.513E−11
	STD	5.617E+01	1.633E+03	1.776E−01	1.847E+03	9.529E+02	7.900E−06	4.125E−10
F2	MEAN	1.099E+07	2.656E+07	4.029E+07	1.575E+08	8.241E+07	9.291E+05	9.677E+05
	STD	7.686E+06	1.460E+07	1.284E+07	4.835E+07	2.705E+07	5.126E+05	5.424E+05
F3	MEAN	7.774E+09	9.170E+09	1.160E+10	3.444E+10	2.025E+10	4.544E+08	1.843E+08
	STD	5.878E+09	6.500E+09	7.181E+09	1.477E+10	8.947E+09	8.172E+08	3.641E+08
F4	MEAN	2.576E+04	3.243E+04	8.954E+04	3.227E+04	3.072E+04	1.153E+03	3.780E+03
	STD	7.089E+03	6.793E+03	2.981E+04	4.564E+03	6.640E+03	8.339E+02	2.212E+03
F5	MEAN	1.197E+01	8.560E+02	8.459E+01	2.155E+03	1.429E+03	5.749E−09	8.335E−11
	STD	2.767E+01	6.282E+02	2.332E+01	6.344E+02	1.012E+03	2.859E−08	5.510E−10
F6	MEAN	7.837E+01	1.728E+02	1.121E+02	7.227E+02	2.956E+02	5.205E+01	4.027E+01
	STD	3.103E+01	8.541E+01	3.585E+01	1.878E+02	8.218E+01	2.583E+01	2.682E+01
F7	MEAN	1.424E+02	6.340E+01	4.855E+02	1.733E+02	1.258E+02	1.206E+02	8.359E+01
	STD	3.434E+01	1.661E+01	8.787E+02	2.766E+01	2.617E+01	3.997E+01	5.229E+01
F8	MEAN	2.095E+01	2.095E+01	2.093E+01	2.094E+01	2.094E+01	2.095E+01	2.095E+01
	STD	5.742E−02	5.188E−02	5.416E−02	4.813E−02	5.697E−02	5.741E−02	5.844E−02
F9	MEAN	3.174E+01	1.840E+01	3.671E+01	3.923E+01	3.787E+01	3.109E+01	2.905E+01
	STD	2.796E+00	3.699E+00	2.945E+00	1.173E+00	1.533E+00	2.855E+00	4.501E+00
F10	MEAN	4.174E+01	4.035E+02	6.632E+01	1.450E+03	7.455E+02	7.442E−01	1.972E−01
	STD	5.033E+01	1.741E+02	2.969E+01	2.349E+02	1.623E+02	3.231E+00	1.406E−01
F11	MEAN	1.300E+02	1.272E+02	4.854E+02	3.603E+02	2.776E+02	2.752E+02	9.948E+01
	STD	3.665E+01	4.367E+01	1.070E+02	3.029E+01	4.232E+01	5.506E+01	3.278E+01
F12	MEAN	2.305E+02	1.648E+02	5.216E+02	3.816E+02	2.980E+02	2.277E+02	1.267E+02
	STD	6.148E+01	6.188E+01	1.244E+02	3.844E+01	2.900E+01	4.900E+01	4.822E+01
F13	MEAN	3.079E+02	2.141E+02	4.845E+02	3.748E+02	2.988E+02	3.019E+02	2.025E+02
	STD	4.875E+01	4.480E+01	7.816E+01	3.067E+01	2.192E+01	6.756E+01	3.664E+01
F14	MEAN	2.830E+03	3.103E+03	5.013E+03	7.088E+03	6.145E+03	3.130E+03	2.518E+03
	STD	6.713E+02	8.829E+02	8.666E+02	3.258E+02	8.664E+02	7.660E+02	1.232E+03
F15	MEAN	4.662E+03	3.495E+03	5.635E+03	7.383E+03	7.251E+03	6.761E+03	6.712E+03
	STD	7.427E+02	1.226E+03	7.637E+02	2.574E+02	3.001E+02	7.688E+02	6.136E+02
F16	MEAN	1.481E+00	2.445E+00	1.649E+00	2.475E+00	2.419E+00	2.426E+00	2.445E+00
	STD	6.087E−01	2.933E−01	4.574E−01	2.865E−01	3.219E−01	2.901E−01	2.783E−01
F17	MEAN	2.158E+02	1.974E+02	5.862E+02	4.902E+02	3.533E+02	2.791E+02	2.201E+02
	STD	5.455E+01	4.372E+01	1.199E+02	3.876E+01	3.499E+01	6.453E+01	4.216E+01
F18	MEAN	3.084E+02	2.781E+02	5.856E+02	4.991E+02	3.535E+02	3.379E+02	2.576E+02
	STD	7.507E+01	5.129E+01	1.323E+02	4.742E+01	3.589E+01	5.394E+01	2.738E+01
F19	MEAN	1.976E+01	4.849E+02	5.135E+01	2.349E+03	4.785E+02	2.102E+02	2.755E+01
	STD	9.640E+00	7.167E+02	1.687E+01	1.422E+03	2.928E+02	2.120E+02	2.400E+01
F20	MEAN	1.339E+01	1.305E+01	1.486E+01	1.393E+01	1.356E+01	1.267E+01	1.261E+01
	STD	7.805E−01	1.650E+00	2.237E−01	3.701E−01	4.071E−01	1.054E+00	6.616E−01
F21	MEAN	3.015E+02	1.173E+03	3.698E+02	1.843E+03	8.958E+02	3.395E+02	3.295E+02
	STD	1.014E+02	4.273E+02	8.791E+01	1.979E+02	2.403E+02	8.930E+01	9.425E+01
F22	MEAN	3.078E+03	3.243E+03	6.136E+03	7.553E+03	6.472E+03	3.284E+03	3.380E+03
	STD	7.038E+02	7.826E+02	1.091E+03	5.651E+02	8.937E+02	6.513E+02	1.855E+03
F23	MEAN	5.418E+03	4.293E+03	6.641E+03	7.800E+03	7.553E+03	6.652E+03	6.738E+03
	STD	8.620E+02	1.476E+03	1.052E+03	3.673E+02	2.652E+02	1.060E+03	8.923E+02
F24	MEAN	2.917E+02	2.547E+02	3.133E+02	3.162E+02	3.017E+02	2.896E+02	2.887E+02
	STD	1.032E+01	1.062E+01	2.021E+01	8.409E+00	5.611E+00	9.862E+00	1.561E+01
F25	MEAN	3.029E+02	2.770E+02	3.228E+02	3.273E+02	3.065E+02	3.001E+02	3.119E+02
	STD	1.141E+01	1.038E+01	1.154E+01	4.568E+00	6.262E+00	9.619E+00	1.602E+01

Table 3 (continued)

No	Index	NNA	GWO	WOA	SCA	JAYA	TLBO	CLJAYA
F26	MEAN	2.042E+02	3.314E+02	3.268E+02	2.115E+02	3.201E+02	3.100E+02	3.308E+02
	STD	2.534E+01	4.896E+01	9.545E+01	3.653E+00	9.241E+01	8.350E+01	7.139E+01
F27	MEAN	1.159E+03	8.406E+02	1.315E+03	1.367E+03	1.284E+03	1.089E+03	1.121E+03
	STD	8.701E+01	7.335E+01	8.760E+01	4.740E+01	4.903E+01	8.871E+01	1.562E+02
F28	MEAN	1.294E+03	1.566E+03	4.451E+03	2.586E+03	1.997E+03	2.715E+03	1.142E+03
	STD	5.902E+02	4.771E+02	9.831E+02	2.094E+02	2.462E+02	7.000E+02	7.888E+02

Table 4 The sorted results of CLJAYA and the compared algorithms on CEC 2013 test suite

Algorithm	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
NNA	4	3	3	3	3	3	5	5	4	3	3	4	5	2
GWO	5	4	4	6	5	5	1	7	1	5	2	2	2	3
WOA	3	5	5	7	4	4	7	1	5	4	7	7	7	5
SCA	7	7	7	5	7	7	6	3	7	7	6	6	6	7
JAYA	6	6	6	4	6	6	4	2	6	6	5	5	3	6
TLBO	2	1	2	1	2	2	3	4	3	2	4	3	4	4
CLJAYA	1	2	1	2	1	1	2	6	2	1	1	1	1	1
Algorithm	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28
NNA	2	2	1	2	3	1	4	1	1	2	4	3	1	4
GWO	3	1	5	1	2	6	3	6	2	1	1	1	7	1
WOA	5	3	2	7	7	3	7	4	5	3	6	6	5	6
SCA	7	7	7	6	6	7	6	7	7	7	7	7	2	7
JAYA	6	6	3	5	5	5	5	5	6	6	5	4	4	5
TLBO	4	5	4	4	4	4	2	3	3	4	3	2	3	2
CLJAYA	1	4	6	3	1	2	1	2	4	5	2	5	6	3

where R_N is the number of independent runs, $\mathbf{x}_{\mathrm{Best},i}$ is the obtained optimal solution at the ith run, and $\mathbf{x}^*$ is the real optimal solution. Besides, Wilcoxon signed-rank test is employed to determine whether there are significance differences between the results obtained by CLJAYA and the compared algorithms on CEC 2013 and CEC 2014 test suites. More specifically, the mean results achieved from 50 independent runs for each algorithm are subjected to this statistical test with a level of significance $\alpha = 0.05$. Tables 5 and 7 show the results produced by Wilcoxon signed-rank test. In Tables 5 and 7, symbol '+' indicates that with 95% certainty the null hypothesis is rejected (p value < 0.05) and CLJAYA outperforms the compared algorithm; symbol '−' means that the null hypothesis is rejected and CLJAYA is inferior to the compared algorithm; symbol '=' demonstrates there is no statistical different between CLJAYA and the compared algorithm (p value ≥ 0.05).

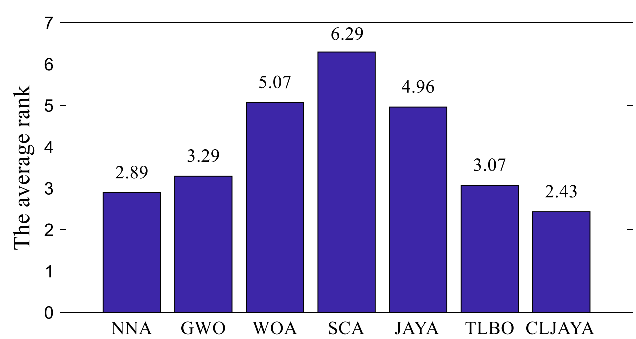

Fig. 3 The average rank of the applied algorithm on CEC 2013 test suite

Benchmark problem set I: CEC 2013 test suite

Table 3 presents the statistical results obtained by CLJAYA and the compared algorithms on CEC 2013 test suite. According to Table 3, CLJAYA can offer the best solutions on nearly half of functions, i.e. F1, F3, F5, F6, F10, F11, F12, F13, F14, F15, F19 and F21. GWO shows a strong competitiveness, which can obtain the best solutions on eight functions, i.e. F7, F9, F16, F18, F24, F25, F26 and F28. Moreover, TLBO, NNA and WOA can find the optimal solutions on two (i.e. F2 and F4), five (i.e.

Table 5 The statistical results produced by Wilcoxon signed−rank test on CEC 2013 test suite

No	CLJAYA vs.											
	NNA		GWO		WOA		SCA		JAYA		TLBO	
	p value	S	*p* value	S	*p* value	S	*p* value	S	*p* value	S	*p* value	S
F1	1.15E−09	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.15E−08	+
F2	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.25E−01	=
F3	9.07E−10	+	8.03E−10	+	8.03E−10	+	7.56E−10	+	7.56E−10	+	1.07E−02	+
F4	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	3.45E−08	−
F5	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	4.77E−08	+
F6	3.20E−07	+	7.56E−10	+	1.30E−09	+	7.56E−10	+	7.56E−10	+	1.69E−02	+
F7	4.56E−07	+	3.33E−02	−	7.56E−10	+	1.07E−08	+	9.18E−06	+	1.20E−04	+
F8	9.88E−01	=	7.61E−01	=	6.18E−02	=	6.06E−01	=	7.17E−01	=	5.92E−01	=
F9	2.40E−03	+	1.09E−09	−	1.47E−09	+	7.56E−10	+	7.56E−10	+	2.03E−02	+
F10	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	4.41E−02	+
F11	4.53E−05	+	6.44E−04	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+
F12	6.38E−09	+	5.85E−03	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	1.20E−08	+
F13	1.76E−09	+	1.26E−01	=	7.56E−10	+	7.56E−10	+	8.53E−10	+	1.50E−08	+
F14	6.88E−02	=	1.07E−02	+	3.57E−09	+	7.56E−10	+	1.15E−09	+	4.61E−03	+
F15	1.15E−09	−	1.02E−09	−	6.99E−08	−	3.09E−08	+	7.16E−07	+	4.09E−01	=
F16	1.20E−08	−	8.81E−01	=	3.78E−09	−	5.02E−01	=	9.88E−01	=	8.06E−01	=
F17	6.75E−01	=	2.08E−02	−	8.03E−10	+	7.56E−10	+	8.03E−10	+	3.85E−06	+
F18	2.49E−04	+	2.48E−02	+	7.56E−10	+	7.56E−10	+	9.07E−10	+	5.69E−09	+
F19	3.41E−02	−	4.79E−08	+	2.41E−06	+	7.56E−10	+	7.56E−10	+	1.09E−09	+
F20	2.98E−05	+	5.53E−02	=	7.56E−10	+	8.03E−10	+	2.66E−09	+	8.89E−01	=
F21	3.67E−01	=	8.53E−10	+	2.48E−03	+	7.56E−10	+	7.56E−10	+	1.36E−01	=
F22	8.51E−01	=	9.81E−01	=	5.95E−08	+	1.15E−09	+	1.34E−08	+	8.89E−01	=
F23	3.92E−07	−	2.93E−08	−	4.54E−01	=	5.37E−09	+	2.34E−08	+	9.65E−01	=
F24	2.57E−01	=	7.56E−10	−	2.48E−08	+	1.13E−08	+	2.21E−05	+	5.53E−01	=
F25	3.29E−03	−	9.07E−10	−	3.89E−04	+	6.17E−07	+	3.02E−02	−	2.31E−05	−
F26	6.63E−08	−	3.84E−02	−	6.31E−02	=	3.45E−08	−	2.45E−01	=	5.59E−01	=
F27	2.15E−01	=	4.51E−09	−	2.12E−07	+	1.47E−09	+	2.60E−07	+	2.29E−01	=
F28	4.84E−01	=	6.44E−04	+	7.56E−10	+	1.09E−09	+	4.54E−08	+	1.13E−08	+
+/=/−		14/8/6		14/5/9		23/3/2		25/2/1		24/3/1		15/11/2

F17, F20, F22, F23 and F27) and one (i.e. F8) functions, respectively.

Besides, SCA and JAYA cannot achieve the optimal solutions on any functions. Based on MEAN from Table 3, Table 4 displays the sorted results obtained by all applied algorithms on CEC2013 test suite according to “tied rank”(Rakhshani and Rahati 2017). Moreover, according to Table 4, Fig. 3 shows the average rank of the applied algorithms. Based on Fig. 3, the applied algorithms can be sorted from best to worst in the following order: CLJAYA, NNA, TLBO, GWO, JAYA, WOA and SCA.

Table 5 displays the Wilcoxon signed-rank test results on CEC 2013 test suite. From Table 5, CLJAYA have a significant advantage over WOA, SCA and JAYA, which can offer better solutions than WOA, SCA and NNA on 23, 25 and 24 functions, respectively. Moreover, NNA, GWO and TLBO is superior to CLJAYA on six (i.e. F15, F16, F19, F23, F25 and F26), nine (i.e. F7, F9, F15, F17, F23, F24, F25, F26, F27 and F28) and two (F4 and F25) functions, respectively. But CLJAYA outperforms NNA, GWO and TLBO on 14 (i.e. F1, F2, F3, F4, F5, F6, F7, F9, F10, F11, F12, F13, F18 and F20), 14 (i.e. F1, F2, F3, F4, F5, F6, F10, F11, F12, F14, F18, F19, F21 and F28) and 15 (i.e. F1, F3, F5, F6, F7, F9, F10, F11, F12, F13, F14, F17, F18, F19 and F28) functions, respectively.

To observe the impact of the designed comprehensive learning mechanism on convergence performance of JAYA, Fig. 4 shows several convergence curves obtained by JAYA and CLJAYA on CEC 2013 test suite. The selected functions are F1, F2, F9, F10, F11, F12, F13, F14, F15, F17, F18, F20, F21, F22, F23, F24, F27 and F28. Obviously, 16 of 18 selected functions are complex multimodal functions.

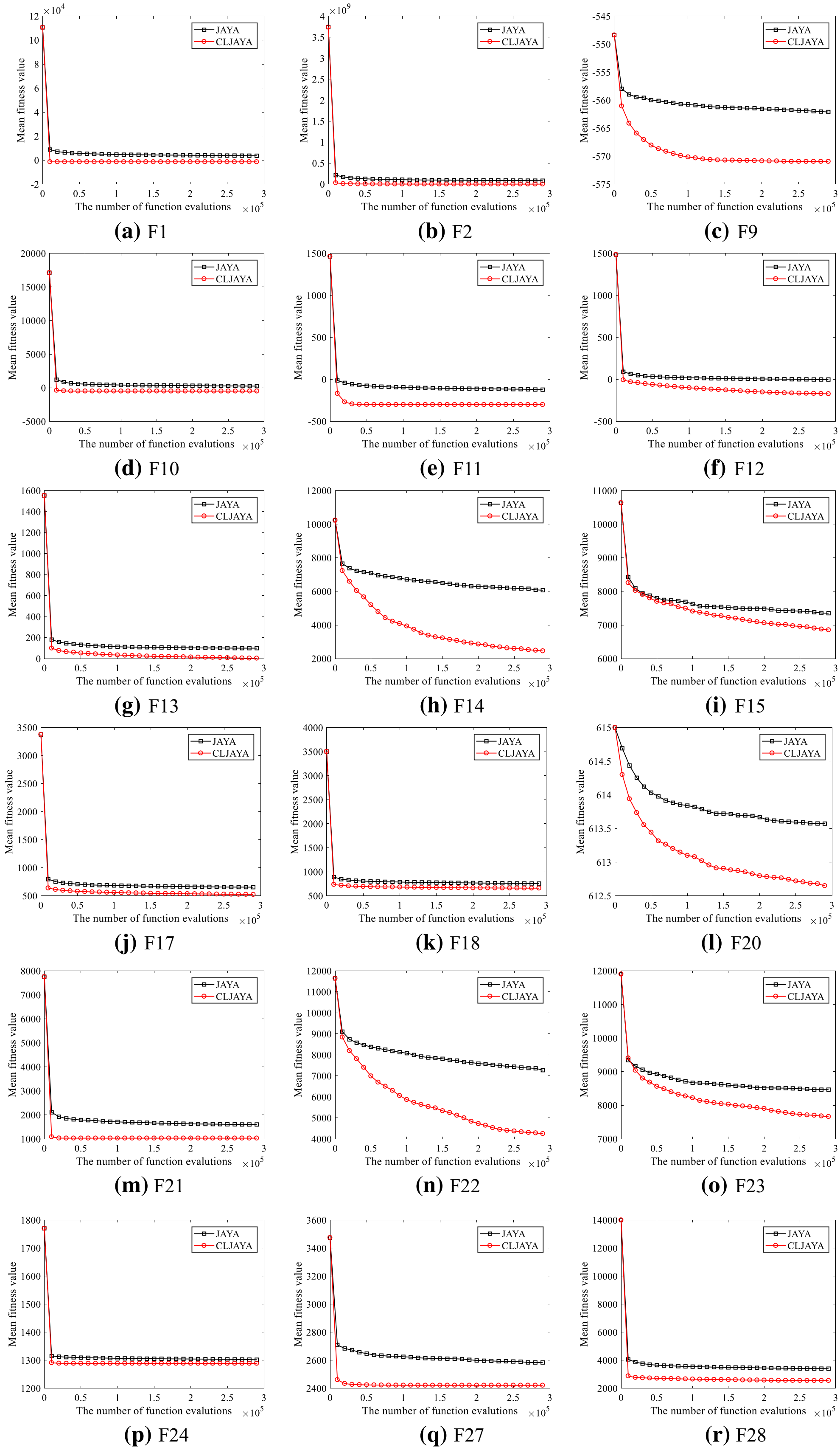

Fig. 4 Several convergence curves obtained by JAYA and CLJAYA on CEC 2013 test suite

Table 6 The statistical results obtained by CLJAYA and the compared algorithms on CEC 2014 test suite

No	Index	NNA	GWO	WOA	SCA	JAYA	TLBO	CLJAYA
F29	MEAN	1.407E+07	7.827E+07	3.084E+07	2.481E+08	5.340E+07	5.054E+05	3.656E+05
	STD	1.133E+07	6.139E+07	1.476E+07	7.456E+07	2.222E+07	6.540E+05	3.009E+05
F30	MEAN	1.184E+07	3.081E+09	9.280E+06	1.694E+10	6.358E+09	2.938E−01	4.930E−02
	STD	3.485E+07	2.060E+09	1.458E+07	3.430E+09	2.075E+09	1.841E+00	1.126E−01
F31	MEAN	1.103E+04	3.182E+04	3.521E+04	3.675E+04	3.781E+04	2.042E+01	7.429E+01
	STD	6.694E+03	9.547E+03	1.911E+04	6.105E+03	1.130E+04	1.020E+02	1.566E+02
F32	MEAN	1.320E+02	3.017E+02	1.950E+02	1.072E+03	4.270E+02	8.209E+01	6.130E+01
	STD	3.876E+01	1.496E+02	5.121E+01	3.165E+02	2.242E+02	3.829E+01	3.321E+01
F33	MEAN	2.024E+01	2.094E+01	2.039E+01	2.094E+01	2.091E+01	2.094E+01	2.090E+01
	STD	2.949E−01	5.272E−02	1.861E−01	4.736E−02	5.495E−02	5.743E−02	8.812E−02
F34	MEAN	2.657E+01	1.538E+01	3.533E+01	3.416E+01	2.977E+01	2.246E+01	1.528E+01
	STD	3.495E+00	3.496E+00	3.444E+00	2.531E+00	5.109E+00	2.595E+00	4.933E+00
F35	MEAN	7.708E−01	3.153E+01	1.030E+00	1.349E+02	1.526E+01	3.546E−01	5.327E−02
	STD	7.307E−01	2.117E+01	7.538E−02	2.386E+01	4.930E+00	7.606E−01	8.880E−02
F36	MEAN	9.006E+01	8.761E+01	1.817E+02	2.424E+02	1.914E+02	1.002E+02	5.512E+01
	STD	2.475E+01	1.861E+01	3.725E+01	1.686E+01	2.671E+01	2.018E+01	1.528E+01
F37	MEAN	1.665E+02	1.107E+02	2.485E+02	2.715E+02	2.481E+02	1.066E+02	7.000E+01
	STD	3.392E+01	2.700E+01	5.405E+01	1.747E+01	2.144E+01	1.976E+01	2.714E+01
F38	MEAN	2.215E+03	2.329E+03	4.033E+03	5.897E+03	4.796E+03	2.507E+03	1.866E+03
	STD	6.426E+02	4.521E+02	8.641E+02	5.014E+02	1.202E+03	6.533E+02	5.125E+02
F39	MEAN	3.951E+03	2.798E+03	4.903E+03	6.972E+03	6.968E+03	3.618E+03	4.634E+03
	STD	5.572E+02	5.803E+02	9.304E+02	2.982E+02	2.882E+02	1.359E+03	1.211E+03
F40	MEAN	8.586E−01	1.641E+00	1.653E+00	2.483E+00	2.403E+00	2.489E+00	2.323E+00
	STD	3.983E−01	1.105E+00	4.948E−01	2.595E−01	2.631E−01	3.046E−01	4.952E−01
F41	MEAN	5.958E−01	5.645E−01	5.132E−01	2.904E+00	1.531E+00	4.760E−01	4.464E−01
	STD	1.058E−01	5.943E−01	1.108E−01	3.748E−01	6.095E−01	1.206E−01	1.144E−01
F42	MEAN	5.135E−01	7.332E+00	2.699E−01	4.368E+01	9.732E+00	3.057E−01	3.077E−01
	STD	2.576E−01	9.834E+00	8.128E−02	7.526E+00	4.692E+00	1.341E−01	1.270E−01
F43	MEAN	3.809E+01	2.390E+02	7.675E+01	3.071E+03	6.438E+01	9.204E+01	2.289E+01
	STD	1.749E+01	5.470E+02	2.366E+01	4.596E+03	1.495E+02	7.172E+01	1.698E+01
F44	MEAN	1.229E+01	1.093E+01	1.245E+01	1.280E+01	1.291E+01	1.115E+01	1.164E+01
	STD	5.326E−01	5.968E−01	7.807E−01	2.508E−01	1.776E−01	7.098E−01	4.678E−01
F45	MEAN	8.637E+05	2.972E+06	4.456E+06	5.823E+06	3.453E+06	8.890E+04	1.123E+05
	STD	1.044E+06	2.951E+06	2.912E+06	2.335E+06	1.780E+06	1.958E+05	1.051E+05
F46	MEAN	6.273E+03	7.145E+06	4.357E+03	1.472E+08	2.612E+07	2.188E+03	6.159E+03
	STD	6.013E+03	1.766E+07	4.301E+03	7.978E+07	4.428E+07	2.879E+03	7.443E+03
F47	MEAN	2.875E+01	5.128E+01	4.419E+01	8.368E+01	3.690E+01	1.827E+01	1.136E+01
	STD	2.643E+01	2.615E+01	3.676E+01	2.115E+01	2.929E+01	1.886E+01	1.394E+01
F48	MEAN	1.263E+04	1.764E+04	2.732E+04	1.446E+04	2.770E+03	2.859E+02	4.027E+02
	STD	7.646E+03	1.016E+04	2.124E+04	5.389E+03	1.589E+03	1.154E+02	2.305E+02
F49	MEAN	2.352E+05	1.040E+06	1.532E+06	1.265E+06	8.264E+05	4.956E+04	5.508E+04
	STD	3.168E+05	1.686E+06	1.559E+06	5.585E+05	8.614E+05	4.488E+04	6.857E+04
F50	MEAN	5.632E+02	4.178E+02	8.143E+02	7.416E+02	5.650E+02	3.786E+02	3.942E+02
	STD	1.758E+02	1.688E+02	2.381E+02	1.556E+02	1.626E+02	1.575E+02	1.645E+02
F51	MEAN	3.165E+02	3.387E+02	3.274E+02	3.618E+02	3.357E+02	3.152E+02	3.153E+02
	STD	1.399E+00	1.607E+01	2.719E+01	1.001E+01	6.206E+00	7.282E−03	2.531E−01
F52	MEAN	2.184E+02	2.000E+02	2.047E+02	2.001E+02	2.562E+02	2.000E+02	2.227E+02
	STD	1.772E+01	4.752E−04	3.328E+00	5.495E−02	1.513E+01	5.021E−04	1.611E+01
F53	MEAN	2.174E+02	2.097E+02	2.146E+02	2.271E+02	2.182E+02	2.003E+02	2.052E+02
	STD	6.904E+00	5.377E+00	1.737E+01	8.987E+00	5.081E+00	2.450E+00	4.435E+00

Table 6 (continued)

No	Index	NNA	GWO	WOA	SCA	JAYA	TLBO	CLJAYA
F54	MEAN	1.007E+02	1.673E+02	1.045E+02	1.023E+02	1.192E+02	1.523E+02	1.045E+02
	STD	1.257E−01	5.486E+01	1.970E+01	6.144E−01	6.358E+01	5.019E+01	1.969E+01
F55	MEAN	7.490E+02	6.910E+02	1.097E+03	7.688E+02	1.039E+03	7.675E+02	8.500E+02
	STD	3.035E+02	1.126E+02	3.403E+02	3.338E+02	1.562E+02	2.773E+02	2.824E+02
F56	MEAN	1.339E+03	1.252E+03	2.271E+03	2.087E+03	1.278E+03	1.510E+03	1.976E+03
	STD	4.077E+02	2.938E+02	6.140E+02	3.351E+02	2.261E+02	4.017E+02	4.843E+02
F57	MEAN	5.454E+05	1.593E+06	7.169E+06	1.155E+07	4.717E+06	6.177E+06	4.467E+07
	STD	2.159E+06	3.318E+06	4.363E+06	6.117E+06	4.914E+06	7.151E+06	4.523E+07
F58	MEAN	1.520E+04	5.475E+04	8.093E+04	2.559E+05	1.461E+04	5.233E+03	1.416E+05
	STD	8.452E+03	5.309E+04	7.031E+04	1.086E+05	1.530E+04	3.550E+03	3.145E+05

Table 7 The sorted results of CLJAYA and the compared algorithms on CEC 2014 test suite

Algorithm	F29	F30	F31	F32	F33	F34	F35	F36	F37	F38	F39	F40	F41	F42	F43
NNA	3	4	3	3	1	4	3	3	4	2	3	1	5	4	2
GWO	6	5	4	5	7	2	6	2	3	3	1	2	4	5	6
WOA	4	3	5	4	2	7	4	5	6	5	5	3	3	1	4
SCA	7	7	6	7	5	6	7	7	7	7	7	6	7	7	7
JAYA	5	6	7	6	4	5	5	6	5	6	6	5	6	6	3
TLBO	2	2	1	2	6	3	2	4	2	4	2	7	2	2	5
CLJAYA	1	1	2	1	3	1	1	1	1	1	4	4	1	3	1

Algorithm	F44	F45	F46	F47	F48	F49	F50	F51	F52	F53	F54	F55	F56	F57	F58
NNA	4	3	4	3	4	3	4	3	5	5	1	2	3	1	3
GWO	1	4	5	6	6	5	3	6	1	3	7	1	1	2	4
WOA	5	6	2	5	7	7	7	4	4	4	3	7	7	5	5
SCA	6	7	7	7	5	6	6	7	3	7	2	4	6	6	7
JAYA	7	5	6	4	3	4	5	5	7	6	5	6	2	3	2
TLBO	2	1	1	2	1	1	1	1	2	1	6	3	4	4	1
CLJAYA	3	2	3	1	2	2	2	2	6	2	4	5	5	7	6

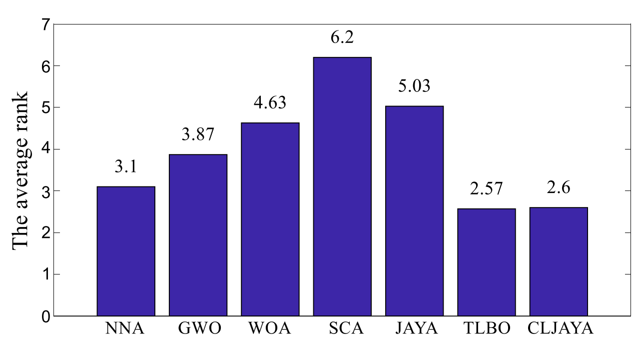

Fig. 5 The average rank obtained by JAYA and CLJAYA on CEC 2014 test suite

As shown in Fig. 4, CLJAYA can find better solutions with faster speed compared with JAYA on these functions, which demonstrates the designed comprehensive learning mechanism can enhance the ability of JAYA to escape from the local optimum.

Benchmark problem set II: CEC 2014 test suite

The statistical results achieved by CLJAYA and the compared algorithms on CEC 2014 test suite are shown in Table 6. From Table 6, CLJAYA can obtain the best solutions on 11 functions, i.e. F29, F30, F32, F34, F35, F36, F37, F38, F41, F43, and F47. TLBO also shows excellent global search ability, which can offer the best solutions on nine functions, i.e. F31, F45, F46, F48, F49, F50, F51, F53 and F58. NNA, GWO, WOA, and SCA can get the best solutions on four (i.e. F33, F40, F54 and F57), five (i.e. F34, F44, F52, F55 and F56), and one (i.e. F42), respectively. SCA and JAYA can't obtain the best solutions on any functions. According to MEAN from Table 6, Table 7 shows the sorted results of "tied rank" obtained by all applied algorithms on

Table 8 The statistical results produced by Wilcoxon signed-rank test on CEC 2014 test suite

No	CLJAYA vs.											
	NNA		GWO		WOA		SCA		JAYA		TLBO	
	p value	S	*p* value	S	*p* value	S	*p* value	S	*p* value	S	*p* value	S
F29	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.25E−01	=
F30	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	3.25E−02	−
F31	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	3.53E−05	−
F32	9.07E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	6.67E−04	+
F33	1.98E−09	−	7.72E−05	+	7.56E−10	−	7.39E−03	+	5.15E−01	=	8.77E−03	+
F34	9.07E−10	+	8.28E−01	=	7.56E−10	+	8.03E−10	+	7.56E−10	+	1.59E−08	+
F35	6.38E−09	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	1.57E−04	+
F36	5.69E−09	+	5.69E−09	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	9.07E−10	+
F37	1.02E−09	+	1.63E−07	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	1.55E−07	+
F38	2.08E−02	+	8.03E−05	+	7.56E−10	+	7.56E−10	+	1.30E−09	+	4.86E−06	+
F39	2.11E−03	−	4.51E−09	−	3.27E−01	=	9.63E−10	+	7.56E−10	+	4.66E−04	−
F40	1.02E−09	−	1.47E−03	−	9.17E−07	−	1.26E−01	=	7.25E−01	=	9.02E−02	=
F41	2.35E−07	+	5.72E−01	=	6.97E−03	+	7.56E−10	+	7.56E−10	+	2.11E−01	=
F42	4.00E−05	+	1.25E−05	+	7.49E−02	=	7.56E−10	+	7.56E−10	+	7.91E−01	=
F43	1.63E−05	+	1.81E−06	+	8.03E−10	+	7.56E−10	+	1.49E−06	+	4.30E−08	+
F44	1.36E−06	+	2.23E−07	−	1.90E−06	+	7.56E−10	+	7.56E−10	+	2.79E−04	−
F45	1.01E−08	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	1.69E−02	−
F46	5.92E−01	=	1.77E−04	+	5.53E−01	=	7.56E−10	+	7.56E−10	+	1.42E−03	−
F47	9.64E−08	+	8.50E−09	+	7.58E−09	+	7.56E−10	+	9.53E−09	+	1.70E−05	+
F48	8.03E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	7.56E−10	+	4.66E−04	−
F49	2.41E−05	+	1.09E−09	+	8.03E−10	+	7.56E−10	+	7.56E−10	+	7.91E−01	=
F50	1.25E−05	+	4.20E−01	=	1.86E−09	+	1.23E−09	+	8.78E−06	+	8.89E−01	=
F51	1.56E−09	+	7.56E−10	+	2.01E−07	+	7.56E−10	+	7.56E−10	+	8.73E−06	+
F52	2.41E−01	=	7.56E−10	−	7.91E−07	−	7.56E−10	−	2.82E−09	+	7.56E−10	−
F53	9.63E−10	+	2.40E−04	+	1.13E−02	+	9.07E−10	+	8.53E−10	+	2.65E−08	−
F54	8.82E−04	+	4.34E−07	+	4.04E−01	=	2.01E−07	+	1.32E−07	+	1.99E−06	+
F55	9.59E−02	=	1.47E−03	−	1.70E−04	+	1.75E−01	=	1.57E−04	+	2.22E−01	=
F56	1.29E−06	−	1.20E−08	−	2.32E−03	+	2.77E−01	=	6.02E−09	−	7.33E−06	−
F57	2.51E−09	−	3.37E−09	−	1.27E−08	−	6.81E−07	−	6.76E−09	−	6.76E−09	−
F58	1.44E−01	=	3.42E−01	=	6.59E−02	=	3.75E−04	+	1.75E−01	=	4.00E−05	−
+/=/−		21/4/5		19/4/7		21/5/4		25/3/2		25/3/2		11/7/12

CEC2014. Moreover, according to Table 7, Fig. 5 gives the average rank of the applied algorithms. As can be seen from Fig. 5, the applied algorithms can be sorted from best to worst in the following order: TLBO, CLJAYA, NNA, GWO, WOA, JAYA and SCA. Although TLBO is the best of the applied algorithms, CLJAYA and TLBO are very close in terms of the average rank, which means CLJAYA and TLBO have the similar performance on CEC 2014 test suite.

The results produced by Wilcoxon signed-rank test for CLJAYA and the compared algorithms on CEC 2014 test suite are displayed in Table 8. According to Table 8, CLJAYA is far superior to NNA, WOA, SCA and JAYA, which can find better solutions than NNA, WOA, SCA and JAYA on 21, 21, 25 and 25 functions, respectively. Moreover, GWO and TLBO can offer better solutions than CLJAYA on seven (i.e. F39, F40, F44, F52, F55, F56 and F57) and 12 (i.e. F30, F31, F39, F44, F45, F46, F48, F52, F53, F56, F57 and F58) functions, respectively. Note that CLJAYA can beat GWO and TLBO on 19 (i.e. F29, F30, F31, F32, F33, F35, F36, F37, F38, F42, F43, F45, F46, F47, F48, F49, F51, F53 and F54) and 11 (i.e. F32, F33, F34, F35, F36, F37, F38, F43, F47, F51 and F54) functions, respectively.

In addition, Fig. 6 shows several convergence curves obtained by JAYA and CLJAYA on CEC2014 test suite to test the effectiveness of the designed comprehensive learning mechanism. The selected functions consist of two unimodal functions (i.e. F29 and F30) and sixteen multimodal

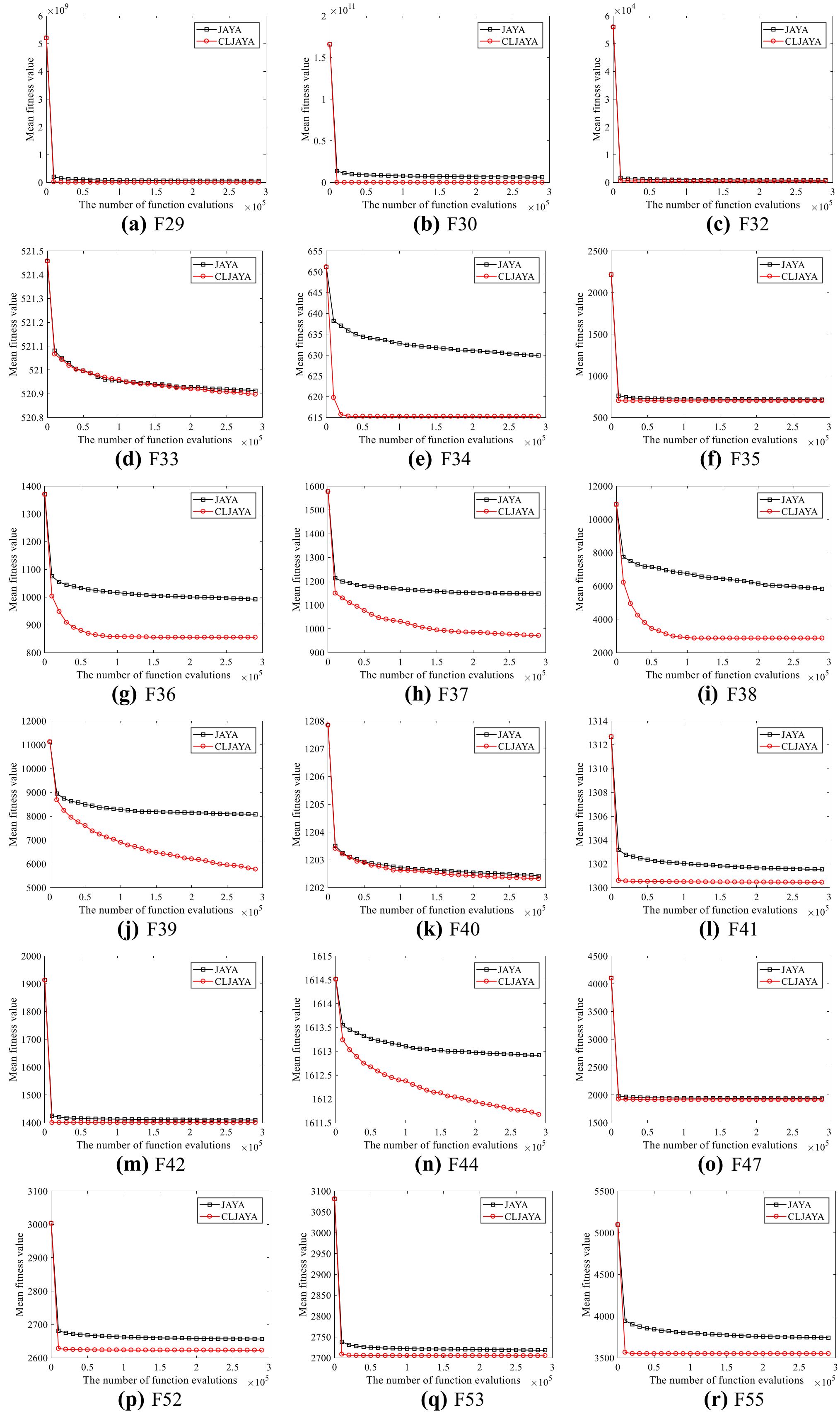

Fig.6 Several convergence curves obtained by JAYA and CLJAYA on CEC2014 test suite

functions (i.e. F32, F33, F34, F35, F36, F37, F38, F39, F40, F41, F42, F44, F47, F52, F53 and F55). From Fig. 6, CLJAYA is superior to JAYA on these functions in terms of solution quality and convergence speed. That is, the designed comprehensive learning mechanism is very effective for enhancing the ability of JAYA escaping from the local optimum.

Discussion for the effectiveness of the improved strategies

In this section, we discuss the effectiveness of the improved strategies based on the experimental results obtained by CLJAYA on CEC 2013 and CEC 2014 test suites.

In order to improve the global search ability of JAYA for complex optimization problems, three learning strategies are designed in CLJAYA. Learning strategy-I is similar with JAYA. In learning strategy-II, mean position is introduced to enhance the chance of CLJAYA to escape from local optima. Learning-strategy-III is to accelerate the convergence speed of CLJAYA, which is guided by the current best solution. In order to study the performance of CLJAYA for complex optimization problems, CEC 2013 and CEC 2014 test suites are employed. Note that the two test suites include 50 multimodal functions and eight unimodal functions, which are very suitable for checking the performance of CLJAYA in solving complex optimization problems.

According to MEAN shown in Tables 3 and 6, JAYA only outperforms CLJAYA on F8, F16, F25, F26, F56, F57 and F58. CLJAYA can beat JAYA on the rest 51 test functions. In addition, Figs. 3 and 4 present the convergence curves obtained by JAYA and CLJAYA for more than 60% of test functions. Based on Fig. 3 and 4, CLJAYA shows better convergence performance than JAYA on these test functions in terms of convergence speed and solution quality. Besides, according to Fig. 3 and 4, JAYA tends to premature convergence while CLJAYA shows strong ability of escaping from the local optima for solving complex optimization problems. Obviously, benefiting from the designed learning strategies, CLJAYA is significant superior to JAYA for the used two test suites in terms of overall optimization performance. More specifically, the designed learning strategies can make full use of population information including the current best solution, the current worst solution, the current mean solution and some current random solutions, which is very helpful for keeping population diversity and enhancing the global search ability of CLJAYA.

Based on the above discussion, the improved strategies introduced to JAYA are very successful and achieve the expected effect.

Applications of CLJAYA on constrained engineering optimization

In this section, CLJAYA is employed to solve five practical engineering design optimization problems. Section 4 has demonstrated the effectiveness of the improved strategies. That is, the improved strategies can significantly enhance the global search ability of JAYA for solving complex optimization problems, which lays a good foundation for using CLJAYA to solve practical complex engineering optimization problems. This section is divided into two parts. Section 5.1 shows the mathematical model of constrained engineering problems and the used mechanism addressing the constrained conditions. Section 5.2 presents the experimental results obtained by CLJAYA for five practical constrained engineering optimization problems.

The mathematical model of constrained engineering problems

Although there are many different types of engineering optimization problems in the real world, their mathematical models all can be formulated as follows:

$$\begin{aligned} &\min f(\mathbf{x}),\ \mathbf{x} = \left(x_1, x_2, \dots, x_D\right)^T \\ &\text{s.t.}\quad h_t(\mathbf{x}) = 0,\ t = 1, 2, \dots, m, \\ &\qquad g_k(\mathbf{x}) \le 0,\ k = 1, 2, \dots, n, \\ &\qquad l_i \le x_i \le u_i, i = 1, 2, \dots, D. \end{aligned} \tag{11}$$

where the objective function is defined by $f(\mathbf{x})$ and $\mathbf{x} = \left(x_1, x_2, \dots, x_D\right)^T$ is a one-dimensional vector of D variables.l_i and u_i are the lower and upper limits of the ith variable, respectively.$h_t(\mathbf{x})$ and $g_k(\mathbf{x})$ are the tth equality constraint and kth inequality constraint, respectively.m and n are the number of equality constraints and inequality constraints, respectively. Moreover, although Eq. (11) describes the minimization problem, the maximization problem can be transformed into minimization one as $-f(\mathbf{x})$. A major barrier in solving a constrained engineering optimization problem is how to handle equality constraints and inequality constraints of the given problem. We transform the constraint optimization problem into an unconstrained optimization problem by the penalty function approach as done in (Gandomi et al. 2011; Gandomi et al. 2013a, b), which can be expressed as

$$P\left(\mathbf{x}, \eta_i, \xi_j\right) = f(\mathbf{x}) + H_i(\mathbf{x}) + G_j(\mathbf{x}) \tag{12}$$

where

$$H_i(\mathbf{x}) = \sum_{i=1}^{p} \eta_i h_i^2(\mathbf{x}) \tag{13}$$

$$G_j(\mathbf{x}) = \sum_{j=1}^{q} \xi_j g_j^2(\mathbf{x}) \tag{14}$$

where $\eta_i\left(1 \le \eta_i\right)$ and $\xi_j\left(0 \le \xi_j\right)$ are penalty factors, and P is the total penalty function. $H_i(\mathbf{x})$ and $G_j(\mathbf{x})$ are the penalty functions of the ith equality constraint and the jth inequality constraint, respectively. The penalty factors (η_i and ξ_j) should be large enough based on the specific optimization problem (Gandomi et al. 2011), which are set to 10e20 in our experiments. As can be seen from Eqs. (12–14), if the ith equality constraint is met,$H_i(\mathbf{x})$ is equal to 0 and has no contribution to P; if the ith equality constraint is violated,$H_i(\mathbf{x})$ will significantly increase and has a significant impact on P. This phenomenon can also happen in $G_j(\mathbf{x})$.

Experiential results on constrained engineering problems

In this section, CLJAYA is used to solve five constrained engineering optimization problems, i.e. welded beam design problem, tension/compression spring design problem, speed reducer design problem, three-bar truss design problem and car impact design problem. In order to show the superiority of CLJAYA for these problems, the obtained results by CLJAYA are compared with those of JAYA and recently reported results. In addition, population size of JAYA and CLJAYA was set to 20 for all test cases. In addition, for every test case, JAYA and CLJAYA were executed 50 independent runs and then the worst value, the mean value, the best value and the standard variance were obtained as shown in Tables 9, 10, 11, 12 and 13. In the five tables, "Worst", "Mean", "Best", "STD", "NFEs" and "NA" stand for the worst value, the mean value, the best value, the standard variance, the number of function evaluations and not available, respectively.

Table 9 The results obtained by CLJAYA and the compared algorithms for welded beam design problem

Algorithm	Worst	Mean	Best	STD	NFEs
CAEP	3.179709	1.971809	1.724852	4.43E−01	50,020
CPSO	1.782143	1.748831	1.728024	1.29E−02	240,000
HPSO	1.814295	1.749040	1.724852	4.01E−02	81,000
SC	6.399678	3.002588	2.385434	9.60E−01	33,095
DE	1.824105	1.768158	1.733461	2.21E−02	204,800
PSO-DE	1.724852	1.724852	1.724852	6.70E−16	66,600
MGA	1.995000	1.919000	1.824500	5.37E−02	NA
WCA	1.744697	1.726427	1.724856	4.29E−03	46,450
QS	1.724852	1.724852	1.724852	NA	20,000
NDE	1.724852	1.724852	1.724852	3.73E−12	8,000
TLNNA	1.724952	1.724866	1.724852	2.09E−05	9,000
DPSO	2.167180	2.067561	2.063119	2.06E−02	40,000
MHS-PCLS	1.724852	1.724852	1.724852	8.11E−11	10,000
εDE-HP	1.724852	1.724852	1.724852	1.40E−12	20,000
hHHO-SCA	3.146846	2.093154	1.779032	3.32E−01	NA
IAFOA	1.724856	1.724856	1.724856	8.99E−07	40,000
MRFO	1.724865	1.724855	1.724852	3.83E−06	30,000
EO	1.736725	1.726482	1.724853	3.36E−03	15,000
UFA	1.724852	1.724852	1.724852	7.96E−11	2,000
I-ABC *greedy*	1.724910	1.724865	1.724852	1.92E−05	14,500
JAYA	1.726289	1.725087	1.724857	3.27E−04	5,000
CLJAYA	1.726242	1.724945	1.724852	2.81E−04	5,000

Case 1: Welded beam design problem

This is a classical engineering design optimization problem, which was proposed by Coello (Coello 2000a, b). Solving this problem is to design a welded beam with the minimum cost. The formula of this problem can be found in Appendix A.1. The optimization constraints of this problem are shear stress (τ), bucking load (P_c), bending stress in the beam (θ) and deflection rate (δ). The design variables of this problem consist of the height of the bar $t(x_1)$, the thickness of the weld $h(x_2)$, the length of the bar $l(x_3)$ and the thickness of the bar $b(x_4)$.

Table 9 shows the results for the welded beam design problem obtained by CAEP (Coello and Becerra 2004), CPSO (Krohling and Coelho 2006), HPSO (Amirjanov 2006), SC (Ray and Liew 2003), DE (Lampinen 2002), PSO-DE (Liu et al. 2010), MGA (Coello 2000a, b), WCA (Eskandar et al. 2012), QS (Zhang et al. 2018), NDE (Mohamed 2018), TLNNA(Zhang et al. 2020), DPSO (Liu et al. 2019), MHS-PCLS (Yi et al. 2019), εDE-HP (Yi et al. 2020), hHHO-SCA (Kamboj et al. 2020), IAFOA (Wu et al. 2018), MRFO (Zhao et al. 2020), EO (Faramarzi et al. 2020), I-ABC *greedy* (Sharma and Abraham 2020), UFA (Brajević and Ignjatović 2019), JAYA and CLJAYA. In Table 9, if one algorithm can get the smallest Best, which means this algorithm can obtain a better solution to design the welded beam than the compared algorithms. According to Table 9, CAEP, HPSO, PSO-DE, QS, NDE, TLNNA, MHS-PCLS, εDE-HP, MRFO, UFA, I-ABC *greedy* and CLJAYA can find the best solution, i.e.1.724852. Note that CLJAYA only consumes 5,000 function evaluations. However, the number of function evaluations consumed by CAEP, HPSO, PSO-DE, QS, NDE, TLNNA, MHS-PCLS, εDE-HP, MRFO and I-ABC *greedy* is 50,020, 81,000, 66,600, 20,000, 8,000, 9,000, 10,000, 20,000, 30,000, and 14,500 respectively. Obviously, CLJAYA has a significant advantage over CAEP, HPSO, PSO-DE, QS, NDE, TLNNA, MHS-PCLS, εDE-HP, MRFO and I-ABC *greedy* in terms of computational efficient. In addition, UFA only needs 2,000 function evaluations, which is more efficient than CLJAYA. Besides, CLJAYA is superior to JAYA on Best, Worst, Mean

Table 10 The results obtained by CLJAYA and the compared algorithms for tension/compression spring design problem

Algorithm	Worst	Mean	Best	STD	NFEs
GA2	0.012822	0.012769	0.012704	3.94E−05	900,000
GA3	0.012973	0.012742	0.012681	5.90E−05	80,000
CPSO	0.012924	0.012730	0.012674	5.20E−04	240,000
HPSO	0.012719	0.012707	0.012665	1.58E−05	81,000
PSO	0.071802	0.019555	0.012857	1.17E−02	2,000
CAEP	0.015116	0.013568	0.012721	8.42E−04	50,020
DE	0.012790	0.012703	0.012670	2.70E−05	204,800
DEDS	0.012738	0.012669	0.012665	1.30E−05	24,000
HEAA	0.012665	0.012665	0.012665	1.40E−09	24,000
PSO-DE	0.012665	0.012665	0.012665	1.20E−08	24,950
PVS	0.012710	0.012670	0.012665	NA	8,000
QS	0.012828	0.012691	0.012665	NA	8,000
NDE(Mohamed 2018)	0.012687	0.012669	0.012665	5.38E−06	24,000
TLNNA	0.012836	0.012689	0.012665	3.24E−05	18,000
DPSO	0.017982	0.013032	0.012665	1.26E−03	30,000
MHS-PCLS	0.012665	0.012665	0.012665	3.61E−11	10,000
εDE-HP	0.012665	0.012665	0.012665	7.80E−10	20,000
hHHO-SCA	0.017774	0.014326	0.012823	1.55E−03	NA
IAFOA	0.012688	0.012673	0.012665	3.02E−06	40,000
MRFO	0.013181	0.012701	0.012676	2.14E−04	50,000
EO	0.013997	0.013017	0.012665	3.91E−04	15,000
DSLC-FOA	0.012692	0.012683	0.012676	6.50E−04	25,000
UFA	0.012665	0.012665	0.012665	3.60E−11	2,000
I-ABC *greedy*	0.018124	0.013731	0.012665	1.12E−06	2,000
JAYA	0.012994	0.012742	0.012677	7.39E−05	6,000
CLJAYA	0.012757	0.012685	0.012665	2.32E−05	6,000

and STD, which means CLJAYA is more suitable for solving the welded beam design problem compared with JAYA.

Case 2: Tension/compression spring design problem

The tension/compression spring design problem is introduced in (Arora 1989). The goal of this problem is to minimize the weight of a tension/compression spring. This problem includes three design variables, i.e. the wire diameter p (x_1), the mean coil diameter D (x_2) and the number of active coils $d(x_3)$. Moreover, four constraints need to be considered. The formula of this problem can be found in Appendix A.2.

Table 10 presents the results for the tension/compression spring design problem obtained by GA2 (Coello 2000a, b), GA3 (Coello and Mezura Montes 2002), CPSO (Krohling and Coelho 2006), HPSO (Liu et al. 2010), PSO (Liu et al. 2010), CAEP (Coello and Becerra 2004), DE (Lampinen 2002), DEDS (Zhang et al. 2008), HEAA (Wang et al. 2009), PSO-DE (Liu et al. 2010), PVS(Savsani and Savsani 2016), QS (Zhang et al. 2018), NDE (Mohamed 2018), TLNNA (Zhang et al. 2020), DPSO (Liu et al. 2019), MHS-PCLS (Yi et al. 2019), εDE-HP (Yi et al. 2020), hHHO-SCA (Kamboj et al. 2020), IAFOA (Wu et al. 2018), MRFO (Zhao et al. 2020), EO(Faramarzi et al. 2020), DSLC-FOA (Du et al. 2018), JAYA, UFA, I-ABC *greedy* and CLJAYA. In Table 10, if one algorithm can achieve the smallest Best, which means this algorithm can offer a better solution to design the tension/compression spring than the compared algorithms. From Table 10, DEDS, HEAA, PSO-DE, PVS, QS, NDE, TLNNA, DPSO, MHS-PCLS, εDE-HP, IAFOA, EO, UFA, I-ABC *greedy* and CLJAYA achieve the optimal objective function value, i.e. 0.012665. The consumed number of function evaluations for DEDS, HEAA, PSO-DE, PVS, QS, NDE, TLNNA, DPSO, MHS-PCLS, εDE-HP, IAFOA, EO and CLJAYA are 24,000, 24,000, 24,950, 8,000, 8,000, 24,000, 18,000, 30,000, 10,000, 20,000, 40,000, 15,000 and 6,000, respectively. Obviously, CLJAYA can find the optimal solution with a faster speed compared with DEDS, HEAA, PSO-DE, PVS, QS, NDE, TLNNA, DPSO, MHS-PCLS, εDE-HP, IAFOA and EO. Note that, UFA and I-ABC *greedy* can consume fewer function evaluations than CLJAYA. In addition, JAYA is inferior to CLJAYA on all considered four indicators.

Table 11 The results obtained by CLJAYA and the compared algorithms for speed reducer design problem

Algorithm	Worst	Mean	Best	STD	NFEs
SC	3009.964736	3001.758264	2994.744241	4.0E+1	54,456
PSO-DE	2996.348204	2996.348174	2996.348167	6.4E−06	54,350
DELC	2994.471066	2994.471066	2994.471066	1.9E−12	30,000
DEDS	2994.471066	2994.471066	2994.471066	3.6E−12	30,000
HEAA	2994.752311	2994.613368	2994.499107	7.0E−02	40,000
FFA	2996.669	2996.51	2996.37	NA	50,000
MBA	2999.65	2996.769	2994.4824	NA	6,300
CSA	3009	3007.1997	3000.98	NA	5,000
PVS	2994.477593	2994.472059	2994.471066	NA	30,000
WCA	2994.505578	2994.474392	2994.471066	7.4E−03	15,150
QS	2994.471066	2994.471066	2994.471066	NA	25,000
NDE	2994.470166	2994.471066	2994.471066	4.17E−12	18,000
TLNNA	2994.474519	2994.471175	2994.471066	5.4E−04	10,500
DPSO	2996.243229	2996.243047	2996.243040	3.4E−05	70,000
MHS-PCLS	2994.471106	2994.471077	2994.471068	7.14E−06	10,000
εDE-HP	2994.471066	2994.471066	2994.471066	6.30E−09	20,000
hHHO-SCA	5053.181732	3696.691485	3029.873076	6.59E+02	NA
IAFOA	2996.348356	2996.348069	2996.347898	3.52E−05	40,000
MRFO	2994.524770	2994.492846	2994.479994	1.46E−02	20,000
UFA	2994.471066	2994.471066	2994.471066	1.53E−08	3,000
I-ABC *greedy*	2994.902	2994.6631	2994.471032	1.87E−12	6,500
JAYA	3033.747875	2996.091421	2994.471066	7.31E−03	7,000
CLJAYA	2994.473148	2994.471151	2994.471066	3.79E−07	7,000

Table. 12 The results obtained by CLJAYA and the compared algorithms for pressure vessel design problem

Algorithm	Worst	Mean	Best	STD	NFEs
SC	263.969756	263.903356	263.895846	1.3E−02	17,610
PSO-DE	263.895843	263.895843	263.895843	4.50E−10	17,600
DSS-MDE	263.895849	263.895843	263.895843	9.72E−07	15,000
MBA	263.915983	263.897996	263.895852	3.93E−03	13,280
DEDS	263.895849	263.895843	263.895843	9.70E−07	15,000
HEAA	263.896099	263.895865	263.895843	4.90E−05	15,000
WCA	263.896201	263.895903	263.895843	8.71E−05	5,250
MFO	NA	NA	263.895980	NA	NA
MVO	NA	NA	263.895850	NA	NA
GOA	NA	NA	263.895881	NA	13,000
SFO	NA	NA	263.895921	NA	NA
PRO	NA	NA	263.895844	NA	NA
hHHO-SCA	264.954366	264.168367	263.895867	3.24E−01	NA
JAYA	263.904386	263.89922	263.89625	2.54E−05	5,000
CLJAYA	263.895844	263.895843	263.895843	3.36E−10	5,000

Case 3: Speed reducer design problem

As a common engineering design problem, the goal of this problem is to minimize the weight of speed reducer subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts, and stresses in the shafts(Brajević and Ignjatović 2019). This problem has seven design variables: the face width $b(x_1)$, module of teeth $m(x_2)$, number of teeth in the pinion $z(x_3)$, length of the first shaft between bearings $l_1(x_4)$, length of the second shaft between bearings $l_2(x_5)$, and the diameter of first $d_1(x_6)$ and second shafts $l_2(x_7)$. Moreover, 11 constraints are included in the problem. Note that, it should be pointed out that this problem has two versions. The only

Table 13 The best solutions obtained by CLJAYA and the compared algorithms for car side impact problem

Variable	PSO	DE	GA	FA	CS	TLBO	TLCS	MHS-PCLS	JAYA	CLJAYA
x_1	0.50000	0.50000	0.50005	0.50000	0.50000	0.50000	0.50000	0.50004	0.50000	0.50000
x_2	1.11670	1.11670	1.28017	1.36000	1.11643	1.11350	1.11630	1.11640	1.11508	1.11634
x_3	0.50000	0.50000	0.50001	0.50000	0.50000	0.50000	0.50000	0.50003	0.50000	0.50000
x_4	1.30208	1.30208	1.03302	1.20200	1.30208	1.30700	1.30230	1.30230	1.30505	1.30224
x_5	0.50000	0.50000	0.50001	0.50000	0.50000	0.50000	0.50000	0.50000	0.50021	0.50000
x_6	1.50000	1.50000	0.50000	1.12000	1.50000	1.50000	1.50000	1.50000	1.50000	1.49999
x_7	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000
x_8	0.34500	0.34500	0.34994	0.34500	0.34500	0.34500	0.34500	0.34499	0.34500	0.34999
x_9	0.19200	0.19200	0.19200	0.19200	0.19200	0.19200	0.19200	0.19215	0.19204	0.19252
x_{10}	− 19.54935	− 19.54935	10.3119	8.87307	8.87307	− 20.0655	− 19.5721	− 19.5690	− 19.7021	− 19.5659
x_{11}	− 0.00431	− 0.00431	0.00167	− 18.99808	− 18.99808	0.11390	0.0157	0.19207	0.89881	− 0.00789
weight	22.84474	22.84298	22.85653	22.84298	22.84294	22.8436	22.8430	22.84361	22.8463	22.84298
NFES	20,000	20,000	20,000	20,000	20,000	20,000	8,000	20,000	20,000	20,000

difference between the two versions is that the limits of the variable x_5(Brajević and Ignjatović 2019; Savsani and Savsani 2016). The variable x_5 lies between 7.3 and 8.3 for the first version while it ranges between 7.8 and 8.3. This experiment is to solve the first version. The formula of this problem can be found in Appendix A.3.

Table 11 displays the results obtained by SC (Ray and Liew 2003), PSO-DE (Liu et al. 2010), DELC (Wang et al. 2009), DEDS (Zhang et al. 2008), HEAA(Wang et al. 2009), FFA (Baykasoğlu and Ozsoydan 2015), MBA (Sadollah et al. 2013), CSA (Askarzadeh 2016), PVS(Savsani and Savsani 2016), WCA (Eskandar et al. 2012), QS (Zhang et al. 2018), NDE (Mohamed 2018), TLNNA (Zhang et al. 2020), DPSO (Liu et al. 2019), MHS-PCLS (Yi et al. 2019), εDE-HP (Yi et al. 2020), hHHO-SCA (Kamboj et al. 2020), IAFOA(Wu et al. 2018), MRFO (Zhao et al. 2020), UFA, I-ABC *greedy*, JAYA and CLJAYA. In Table 11, one algorithm with the smallest Best can give a better solution to design the speed reducer than the compared algorithms. As presented in Table 11 I-ABC *greedy* is the best, which can find the optimal solution, i.e. 2994.471032. In addition, DELC, DEDS, WCA, QS, PVS, NDE, TLNNA, εDE-HP, UFA, JAYA and CLJAYA can find the same optimal solution, i.e. 2994.471066. Note that CLJAYA is superior to JAYA on the rest indicators including Worst, Mean and STD, which indicates CLJAYA has a better comprehensive performance than JAYA for speed reducer design problem.

Case 4: Three-bar truss design problem

The goal of this problem is to minimize the volume of a statistically loaded three-bar truss subject to stress constraints on each of the truss members by adjusting cross sectional areas (x_1, x_2). Moreover, three constraints also need to be taken into account. The formula of this problem can be found in Appendix A.4.

Table 12 displays the results for the three-bar truss design problem obtained by SC (Ray and Liew 2003), PSO-DE (Liu et al. 2010), DSS-MDE (Zhang et al. 2008), MBA(Sadollah et al. 2013), DEDS (Zhang et al. 2008), HEAA (Wang et al. 2009), WCA (Eskandar et al. 2012), MFO (Mirjalili 2015), MVO(Mirjalili et al. 2016), GOA (Saremi et al. 2017), SFO (Shadravan et al. 2019), PRO (Samareh Moosavi and Bardsiri 2019), hHHO-SCA (Kamboj et al. 2020), JAYA and CLJAYA. In Table 12, one algorithm with the smallest Best can find a better solution to design the three-bar truss than the compared algorithms. As can be seen from Table 12, PSO-DE, DSS-MDE, DEDS, HWAA, WCA, CLJAYA can offer the optimal solution. By observing Table 12, PSO-DE, DSS-MDE, DEDS and HEAA consume more than 10,000 function evaluations while CLJAYA only consumes 5,000 function evaluations. Moreover, WCA shows strong competitiveness with 5,250 function evaluations. Note that CLJAYA is slight superior to WCA in terms of convergence speed. Besides, CLJAYA outperforms JAYA on Worst, Mean, Best and STD.

Case 5: Car impact design problem

The design problem of car side impact is stated by Gu et al. (Gu et al. 2001). On the foundation of European Enhanced Vehicle-Safety Committee procedures, a car is exposed to a side-impact (Gandomi et al. 2011). The goal of this case is to minimize the weight related to nine influence parameters including thickness of B-Pillar inner, B-Pillar reinforcement, floor side inner, cross members, door beam, door beltline reinforcement and roof rail $(x_1 – x_7)$, materials of B-Pillar inner and floor side inner $(x_8 - x_9)$ and barrier height and hitting position $(x_{10} - x_{11})$. Moreover, ten inequality constraints associated with the car side impact design problem need to be considered. The formula of this problem can be found in Appendix A.5.

Table 13 gives the best solutions obtained by PSO (Gandomi et al. 2013a, b), DE (Gandomi et al. 2013a, b), GA (Gandomi et al. 2013a, b), FA (Gandomi et al. 2011), CS (Gandomi et al. 2013a, b), TLBO (Huang et al. 2015), TLCS (Huang et al. 2015), MHS-PCLS (J. Yi et al. 2019), JAYA and CLJAYA. In Table 13, the algorithm with the smallest weight can find the better solution to the car impact design problem compared with the other algorithms. From Table 13, CS achieves the optimal solution. Note that DE, FA and CLJAYA can offer very competitive solutions.

Conclusions and further work

This paper presents an improved Jaya algorithm called comprehensive learning Jaya algorithm (CLJAYA) for solving engineering optimization problems. The proposed CLJAYA has a very simple structure and only depends on the essential population size and terminal condition for solving optimization problems. In CLJAYA, the designed comprehensive learning mechanism with three different learning strategies is introduced to enhance its ability of escaping from the local optimum. In order to investigate the effectiveness of the improved strategies, CLJAYA is used to solve CEC 2013 test suite, CEC 2014 test suite and five challenging engineering optimization problems. In addition, CEC 2013 and CEC 2014 test suits have 58 test functions. Note that 50 of 58 test functions are multimodal functions. Besides, the used five engineering optimization problems need to meet the given complex constraints. Thus, these test cases are very suitable for examining the performance of CLJAYA for complex optimization problems. The experimental results prove the superiority of CLJAYA in solving complex optimization problems by comparing with JAYA and some state-of-the-art algorithms in terms of solution quality and computational efficiency. In other words, the designed strategies for improving JAYA are very effective.

Further work will focus on the following two aspects. On the one hand, CLJAYA is a metaheuristic algorithm without special parameters and has a great potential to be widely used. Thus, we intend to use CLJAYA to solve more practical engineering optimization problems, such as flexible job-shop scheduling problem and vehicle routing problem with time windows. On another hand, we discuss the advantages of metaheuristic algorithms without special parameters in this work. However, most of the reported metaheuristic algorithms have special parameters. Note that developing metaheuristic algorithms without special parameters has not been regarded highly by researchers. Thus, we will develop more metaheuristic algorithms without special parameters to solve different types of optimization problems in the future research.

Compliance with ethical standards

Competing interest The authors declare that there is no conflict of interests regarding the publication of this paper.

Appendix A

A.1 welded beam design problem

$$
\begin{aligned}
&\text{Minimize } f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4\left(14 + x_2\right)\\
&\text{Subject to:}\\
&g_1(x) = \tau(x) - \tau_{\max} \le 0\\
&g_2(x) = \sigma(x) - \sigma_{\max} \le 0\\
&g_3(x) = x_1 - x_4 \le 0\\
&g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4\left(14 + x_2\right) - 5 \le 0\\
&g_5(x) = 0.125 - x_1 \le 0\\
&g_6(x) = \delta(x) - \delta_{\max} \le 0\\
&g_7(x) = P - P_c(x) \le 0\\
&0.1 \le x_i \le 2 \;\; i = 1,4\\
&0.1 \le x_i \le 10 \; i = 2,3\\
&\text{where,}\\
&\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}, \tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right), R = \sqrt{\left(\frac{x_2}{2}\right)^2 + \left(\frac{x_1+x_3}{2}\right)^2},\\
&J = 2\left(\sqrt{2}x_1x_2\left(\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right)\right), \sigma(x) = \frac{6PL}{x_4x_3^2}, \delta(x) = \frac{4PL^3}{Ex_3^3x_4}, P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)\\
&P = 6000\text{lb}, L = 14\text{in}, E = 30 \times 10^6\text{psi}, G = 12 \times 10^6\text{psi}, \tau_{\max} = 13,600\text{psi}, \; \sigma_{\max} = 30,000\text{psi}, \delta_{\max} = 0.25\text{in}
\end{aligned}
$$

A.2 Tension/compression spring design problem

Minimize $f(x) = (x_3 + 2)x_2 x_1^2$

Subject to:

$g_1(x) = 1 - \frac{x_2^3 x_3}{71{,}785 x_1^4} \le 0$

$g_2(x) = 4x_2^2 - \frac{x_1 x_2}{12.566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \le 0$

$g_3(x) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0$

$g_4(x) = x_2 + \frac{x_1}{1.5} - 1 \le 0$

where,

$0.05 \le x_1 \le 2,\ 0.25 \le x_2 \le 1.30,\ 2.00 \le x_3 \le 15.00$

A.3 Speed reducer design problem

Minimize $f(x) = 0.7854 x_1 x_2^2 (3.3333 x_3^2 + 14.9334 x_3 - 43.0934) - 1.508 x_1 (x_6^2 + x_7^2) + 7.4777 (x_6^3 + x_7^3) + 0.7854 (x_4 x_6^2 + x_5 x_7^2)$

Subject to:

$g_1(x) = \frac{27}{x_1 x_2^2 x_3} - 1 \le 0$

$g_2(x) = \frac{397.5}{x_1 x_2^2 x_3} - 1 \le 0$

$g_3(x) = \frac{1.93 x_4^3}{x_2 x_6^4 x_3} - 1 \le 0$

$g_4(x) = \frac{1.93 x_5^3}{x_2 x_7^4 x_3} - 1 \le 0$

$g_5(x) = \frac{\left(\left(\frac{745 x_4}{x_2 x_3}\right)^2 + 16.9 \times 10^6\right)^{1/2}}{110 x_6^3} - 1 \le 0$

$g_6(x) = \frac{\left(\left(\frac{745 x_5}{x_2 x_3}\right)^2 + 157.5 \times 10^6\right)^{1/2}}{85 x_7^3} - 1 \le 0$

$g_7(x) = \frac{x_2 x_3}{40} - 1 \le 0$

$g_8(x) = \frac{5 x_2}{x_1} - 1 \le 0$

$g_9(x) = \frac{x_1}{12 x_2} - 1 \le 0$

$g_{10}(x) = \frac{1.5 x_6 + 1.9}{x_4} - 1 \le 0$

$g_{11}(x) = \frac{1.1 x_7 + 1.9}{x_5} - 1 \le 0$

where,

$2.6 \le x_1 \le 3.6,\ 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28,\ 7.3 \le x_4 \le 8.3, 7.3 \le x_5 \le 8.3,\ 2.9 \le x_6 \le 3.9,\ 5.0 \le x_7 \le 5.5$

A.4 Three-bar truss design problem

Minimize $f(x) = (2\sqrt{2} x_1 + x_2) \times l$

Subject to:

$g_1(x) = \frac{\sqrt{2} x_1 + x_2}{\sqrt{2} x_1^2 + 2 x_1 x_2} P - \sigma \le 0$

$g_2(x) = \frac{x_2}{\sqrt{2} x_1^2 + 2 x_1 x_2} P - \sigma \le 0$

$g_3(x) = \frac{1}{\sqrt{2} x_1^2 + 2 x_1 x_2} P - \sigma \le 0$

where,

$0 \le x_i \le 1,\ i = 1, 2$

$l = 100\text{cm}, P = 2\text{k}N/\text{cm}^2, \sigma = 2\text{k}N/\text{cm}^2$

A.5 Car impact design problem

$$\text{Minimize } f(\mathbf{x}) = 1.98 + 4.90x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 2.73x_7$$

Subject to

$$g_1(\mathbf{x}) = 1.16 - 0.3717x_2x_4 - 0.00931x_2x_{10} - 0.484x_3x_9 + 0.01343x_6x_{10} \leq 1\text{KN}$$
$$g_2(\mathbf{x}) = 0.261 - 0.0159x_1x_2 - 0.0188x_1x_8 - 0.0191x_2x_7 + 0.0144x_3x_5 + 0.0008757x_5x_{10} + 0.08045x_6x_9 + 0.00139x_8x_{11} + 0.00001575x_{10}x_{11} \leq 0.32 \text{ m/s}$$
$$g_3(\mathbf{x}) = 0.214 + 0.00817x_5 - 0.131x_1x_8 - 0.0704x_1x_9 + 0.03099x_2x_6 - 0.018x_2x_7 + 0.0208x_3x_8 + 0.121x_3x_9 - 0.00364x_5x_6 + 0.0007715x_5x_{10} - 0.0005354x_6x_{10} + 0.00121x_8x_{11} \leq 0.32 \text{ m/s}$$
$$g_4(\mathbf{x}) = 0.74 - 0.61x_2 - 0.163x_3x_8 + 0.001232x_3x_{10} - 0.166x_7x_9 + 0.227x_2^2 \leq 0.32 \text{ m/s}$$
$$g_5(\mathbf{x}) = 28.98 + 3.818x_3 - 4.2x_1x_2 + 0.0207x_5x_{10} + 6.63x_6x_9 - 7.7x_7x_8 + 0.32x_9x_{10} \leq 32 \text{ mm}$$
$$g_6(\mathbf{x}) = 33.86 + 2.95x_3 + 0.1792x_{10} - 5.057x_1x_2 - 11.0x_2x_8 - 0.0215x_5x_{10} - 9.98x_7x_8 + 22.0x_8x_9 \leq 32 \text{ mm}$$
$$g_7(\mathbf{x}) = 46.36 - 9.9x_2 - 12.9x_1x_8 - 5.057x_1x_2 + 0.1107x_3x_{10} \leq 32 \text{ mm}$$
$$g_8(\mathbf{x}) = 4.72 - 0.5x_4 - 0.19x_2x_3 - 0.0122x_4x_{10} + 0.009325x_6x_{10} + 0.000191x_{11}^2 \leq 4\text{KN}$$
$$g_9(\mathbf{x}) = 10.58 - 0.674x_1x_2 - 1.95x_2x_8 + 0.02054x_3x_{10} - 0.0198x_4x_{10} + 0.028x_6x_{10} \leq 9.9 \text{ mm/ms}$$
$$g_{10}(\mathbf{x}) = 16.45 - 0.489x_3x_7 - 0.843x_5x_6 + 0.0432x_9x_{10} - 0.0556x_9x_{11} - 0.000786x_{11}^2 \leq 15.7 \text{ mm/ms}$$

where $0.5 \leq x_i \leq 1.5,\ i = 1,2,3,4,5,6,7;\ 0.192 \leq x_i \leq 0.345,\ i = 8,9;\ -30 \leq x_i \leq 30,\ i = 10,11.$

References

Amirjanov, A. (2006). The development of a changing range genetic algorithm. *Computer Methods in Applied Mechanics and Engineering, 195*(19), 2495–2508. https://doi.org/10.1016/j.cma.2005.05.014.

Arora, J. S. (1989). *Introduction to optimum design*. New York: McGraw-Hill.

Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Computers & Structures, 169,* 1–12. https://doi.org/10.1016/j.compstruc.2016.03.001.

Baykasoğlu, A., & Ozsoydan, F. B. (2015). Adaptive firefly algorithm with chaos for mechanical design optimization problems. *Applied Soft Computing, 36,* 152–164. https://doi.org/10.1016/j.asoc.2015.06.056.

Brajević, I., & Ignjatović, J. (2019). An upgraded firefly algorithm with feasibility-based rules for constrained engineering optimization problems. *Journal of Intelligent Manufacturing, 30*(6), 2545–2574. https://doi.org/10.1007/s10845-018-1419-6.

Cheng, M.-Y., & Prayogo, D. (2014). Symbiotic organisms search: A new metaheuristic optimization algorithm. *Computers & Structures, 139,* 98–112. https://doi.org/10.1016/j.compstruc.2014.03.007.

Coello, C. A. C. (2000). Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering Systems, 17*(4), 319–346.

Coello, C. A. C., & Becerra, R. L. (2004). Efficient evolutionary optimization through the use of a cultural algorithm. *Engineering Optimization, 36*(2), 219–236.

Coello Coello, C. A. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry, 41*(2), 113–127. https://doi.org/10.1016/S0166-3615(99)00046-9.

Coello Coello, C. A., & Mezura Montes, E. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics, 16*(3), 193–203. https://doi.org/10.1016/S1474-0346(02)00011-3.

Du, T.-S., Ke, X.-T., Liao, J.-G., & Shen, Y.-J. (2018). DSLC-FOA : Improved fruit fly optimization algorithm for application to structural engineering design optimization problems. *Applied Mathematical Modelling, 55,* 314–339. https://doi.org/10.1016/j.apm.2017.08.013.

Eskandar, H., Sadollah, A., Bahreininejad, A., & Hamdi, M. (2012). Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers & Structures, 110–111,* 151–166. https://doi.org/10.1016/j.compstruc.2012.07.010.

Faramarzi, A., Heidarinejad, M., Stephens, B., & Mirjalili, S. (2020). Equilibrium optimizer: A novel optimization algorithm. *Knowledge-Based Systems, 191,* 105190. https://doi.org/10.1016/j.knosys.2019.105190.

Gandomi, A. H., Yang, X.-S., & Alavi, A. H. (2011). Mixed variable structural optimization using Firefly Algorithm. *Computers & Structures, 89*(23), 2325–2336. https://doi.org/10.1016/j.compstruc.2011.08.002.

Gandomi, A. H., Yang, X.-S., & Alavi, A. H. (2013a). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engineering with Computers, 29*(1), 17–35. https://doi.org/10.1007/s00366-011-0241-y.

Gandomi, A. H., Yang, X.-S., Alavi, A. H., & Talatahari, S. (2013b). Bat algorithm for constrained optimization tasks. *Neural Computing and Applications, 22*(6), 1239–1255. https://doi.org/10.1007/s00521-012-1028-9.

Gu, L., Yang, R., Tho, C.-H., Makowski, M., Faruque, O., & Li, Y. (2001). Optimization and robustness for crashworthiness of side impact. *International journal of vehicle design, 26*(4), 348–360.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Oxford, England: U Michigan Press.

Huang, J., Gao, L., & Li, X. (2015). An effective teaching-learning-based cuckoo search algorithm for parameter optimization problems in structure designing and machining processes. *Applied Soft Computing, 36,* 349–356. https://doi.org/10.1016/j.asoc.2015.07.031.

Li, J., Zhang, J., Jiang, C., & Zhou, M. (2015). Composite particle swarm optimizer with historical memory for function optimization. *IEEE Transactions on Cybernetics, 45*(10), 2350–2363. https://doi.org/10.1109/TCYB.2015.2424836.

Kamboj, V. K., Nandi, A., Bhadoria, A., & Sehgal, S. (2020). An intensify Harris Hawks optimizer for numerical and engineering optimization problems. *Applied Soft Computing, 89,* 106018. https://doi.org/10.1016/j.asoc.2019.106018.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks* (Vol. 4, pp. 1942–1948). IEEE.

Krohling, R. A., & dos Santos Coelho, L. (2006). Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 36*(6), 1407–1416. https://doi.org/10.1109/TSMCB.2006.873185.

K.S., S. R., & Murugan, S. (2017). Memory based Hybrid Dragonfly Algorithm for numerical optimization problems. *Expert Systems with Applications, 83,* 63–78. https://doi.org/10.1016/j.eswa.2017.04.033.

Lampinen, J. (2002). A constraint handling approach for the differential evolution algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)* (Vol. 2, pp. 1468–1473 vol.2). https://doi.org/10.1109/CEC.2002.1004459

Lee, K. S., & Geem, Z. W. (2004). A new structural optimization method based on the harmony search algorithm. *Computers & Structures, 82*(9), 781–798. https://doi.org/10.1016/j.compstruc.2004.01.002.

Lee, K. S., & Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering, 194*(36), 3902–3933. https://doi.org/10.1016/j.cma.2004.09.007.

Liu, H., Wang, Y., Tu, L., Ding, G., & Hu, Y. (2019). A modified particle swarm optimization for large-scale numerical optimizations and engineering design problems. *Journal of Intelligent Manufacturing, 30*(6), 2407–2433. https://doi.org/10.1007/s10845-018-1403-1.

Liu, H., Cai, Z., & Wang, Y. (2010). Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing, 10*(2), 629–640. https://doi.org/10.1016/j.asoc.2009.08.031.

Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89,* 228–249. https://doi.org/10.1016/j.knosys.2015.07.006.

Mirjalili, S. (2016). SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowledge-Based Systems, 96,* 120–133. https://doi.org/10.1016/j.knosys.2015.12.022.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95,* 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008.

Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2016). Multi-Verse Optimizer: a nature-inspired algorithm for global optimization. *Neural Computing and Applications, 27*(2), 495–513. https://doi.org/10.1007/s00521-015-1870-7.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software, 69,* 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007.

Mohamed, A. W. (2018). A novel differential evolution algorithm for solving constrained engineering optimization problems. *Journal of Intelligent Manufacturing, 29*(3), 659–692. https://doi.org/10.1007/s10845-017-1294-6.

Rakhshani, H., & Rahati, A. (2017). Snap-drift cuckoo search: A novel cuckoo search optimization algorithm. *Applied Soft Computing, 52,* 771–794. https://doi.org/10.1016/j.asoc.2016.09.048.

Cheng, R., & Jin, Y. (2015). A competitive swarm optimizer for large scale optimization. *IEEE Transactions on Cybernetics, 45*(2), 191–204. https://doi.org/10.1109/TCYB.2014.2322602.

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

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

Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2012). Teaching–learning-based optimization: An optimization method for continuous non-linear large scale problems. *Information Sciences, 183*(1), 1–15. https://doi.org/10.1016/j.ins.2011.08.006.

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Special Section on High Order Fuzzy Sets, 179*(13), 2232–2248. https://doi.org/10.1016/j.ins.2009.03.004.

Ray, T., & Liew, K. M. (2003). Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation, 7*(4), 386–396. https://doi.org/10.1109/TEVC.2003.814902.

Sadollah, A., Bahreininejad, A., Eskandar, H., & Hamdi, M. (2013). Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Applied Soft Computing, 13*(5), 2592–2612. https://doi.org/10.1016/j.asoc.2012.11.026.

Sadollah, A., Sayyaadi, H., & Yadav, A. (2018). A dynamic metaheuristic optimization model inspired by biological nervous systems: Neural network algorithm. *Applied Soft Computing, 71,* 747–782. https://doi.org/10.1016/j.asoc.2018.07.039.

Samareh Moosavi, S. H., & Bardsiri, V. K. (2019). Poor and rich optimization algorithm: A new human-based and multi populations algorithm. *Engineering Applications of Artificial Intelligence, 86,* 165–181. https://doi.org/10.1016/j.engappai.2019.08.025.

Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: Theory and application. *Advances in Engineering Software, 105,* 30–47. https://doi.org/10.1016/j.advengsoft.2017.01.004.

Savsani, P., & Savsani, V. (2016). Passing vehicle search (PVS): A novel metaheuristic algorithm. *Applied Mathematical Modelling, 40*(5), 3951–3978. https://doi.org/10.1016/j.apm.2015.10.040.

Shadravan, S., Naji, H. R., & Bardsiri, V. K. (2019). The sailfish optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. *Engineering Applications of Artificial Intelligence, 80,* 20–34. https://doi.org/10.1016/j.engappai.2019.01.001.

Sharma, T. K., & Abraham, A. (2020). Artificial bee colony with enhanced food locations for solving mechanical engineering design problems. *Journal of Ambient Intelligence and Humanized Computing, 11*(1), 267–290. https://doi.org/10.1007/s12652-019-01265-7.

Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization, 11*(4), 341–359.

Tanweer, M. R., Suresh, S., & Sundararajan, N. (2016). Dynamic mentoring and self-regulation based particle swarm optimization algorithm for solving complex real-world optimization problems. *Information Sciences, 326,* 1–24. https://doi.org/10.1016/j.ins.2015.07.035.

Wang, Y., Cai, Z., Zhou, Y., & Fan, Z. (2009). Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. *Structural and Multidisciplinary Optimization, 37*(4), 395–413. https://doi.org/10.1007/s00158-008-0238-3.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67–82. https://doi.org/10.1109/4235.585893.

Wu, L., Liu, Q., Tian, X., Zhang, J., & Xiao, W. (2018). A new improved fruit fly optimization algorithm IAFOA and its application to solve engineering optimization problems. *Knowledge-Based Systems, 144,* 153–173. https://doi.org/10.1016/j.knosys.2017.12.031.

Xiang, Z., Ji, D., Zhang, H., Wu, H., & Li, Y. (2019). A simple PID-based strategy for particle swarm optimization algorithm. *Information Sciences, 502,* 558–574. https://doi.org/10.1016/j.ins.2019.06.042.

Yang, X.-S., & Deb, S. (2014). Cuckoo search: recent advances and applications. *Neural Computing and Applications, 24*(1), 169–174. https://doi.org/10.1007/s00521-013-1367-1.

Yi, J., Li, X., Chu, C.-H., & Gao, L. (2019). Parallel chaotic local search enhanced harmony search algorithm for engineering design optimization. *Journal of Intelligent Manufacturing, 30*(1), 405–428. https://doi.org/10.1007/s10845-016-1255-5.

Yi, W., Gao, L., Pei, Z., Lu, J., & Chen, Y. (2020). ε Constrained differential evolution using halfspace partition for optimization problems. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-020-01565-2.

Zhang, J., Xiao, M., Gao, L., & Pan, Q. (2018). Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems. *Applied Mathematical Modelling, 63,* 464–490. https://doi.org/10.1016/j.apm.2018.06.036.

Zhang, K., Huang, Q., & Zhang, Y. (2019). Enhancing comprehensive learning particle swarm optimization with local optima topology. *Information Sciences, 471,* 1–18. https://doi.org/10.1016/j.ins.2018.08.049.

Zhang, M., Luo, W., & Wang, X. (2008). Differential evolution with dynamic stochastic selection for constrained optimization. *Nature Inspired Problem-Solving, 178*(15), 3043–3074. https://doi.org/10.1016/j.ins.2008.02.014.

Zhang, Y., Jin, Z., & Chen, Y. (2020). Hybrid teaching–learning-based optimization and neural network algorithm for engineering design optimization problems. *Knowledge-Based Systems, 187,* 104836. https://doi.org/10.1016/j.knosys.2019.07.007.

Zhao, W., Zhang, Z., & Wang, L. (2020). Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. *Engineering Applications of Artificial Intelligence, 87,* 103300. https://doi.org/10.1016/j.engappai.2019.103300.