ATLANTIS PRESS International Journal of Computational Intelligence Systems, Vol. 10 (2017) 1102–1115

An Enhanced Jaya Algorithm with a Two Group Adaption

Chibing Gong

Department of Computer Information Engineering,
Guangdong Technical College of Water Resource and Electrical Engineering,
Guangzhou 510635, China
E-mail:spencergong@qq.com

Received 4 February 2017

Accepted 13 July 2017

Abstract

This paper proposes a novel performance enhanced Jaya algorithm with a two group adaption (E-Jaya). Two improvements are presented in E-Jaya. First, instead of using the best and the worst values in Jaya algorithm, E-Jaya separates all candidates into two groups: the better and the worse groups based on their fitness values, then the mean of the better group and the mean of the worse group are used. Second, in order to add non algorithm-specific parameters in E-Jaya, a novel adaptive method of dividing the two groups has been developed. Finally, twelve benchmark functions with different dimensionality, such as 40, 60, and 100, were evaluated using the proposed E-Jaya algorithm. The results show that E-Jaya significantly outperformed Jaya algorithm in terms of the solution accuracy. Additionally, E-Jaya was also compared with a differential evolution (DE), a self-adapting control parameters in differential evolution (jDE), a firefly algorithm (FA), and a standard particle swarm optimization 2011 (SPSO2011) algorithm. E-Jaya algorithm outperforms all the algorithms.

Keywords: Jaya algorithm, E-Jaya algorithm, parameter adaption, global optimization, heuristic algorithms

1. Introduction

The past three decades has given way to many population based intelligent algorithms, inspired by biological behaviors. These algorithms have proposed solutions to several practical and complex global optimization problems. An ant colony optimization algorithm[1] (ACO) was brought forth in 1992, based on observations of the characteristics of ants seeking food. Particle swarm optimization[2] (PSO) is an algorithm that imitates flocks of birds flying during migration. The differential evolution[3] (DE) was proposed by simulating the natural adaptation of biological species. A firefly algorithm[4] (FA) was proposed, inspired from the behaviors of firefly with the flashing characteristics, and finally a bat algorithm[5] (BA) was proposed which was based on the echolocation principle of microbats.

Jaya[6], a population based intelligent algorithm, is a simple, powerful optimization algorithm designed to approximate optimal solutions. This algorithm is based on the idea that the optimal solution should focus on the best solution, while avoiding the worst solution. The advantage of Jaya algorithm is that it dictates the general control parameters, which are the number of generations and population size, and no algorithm-specific parameters, which are special parameters to control for different algorithms. Thus, it is only necessary to simply choose the common control parameters in the Jaya algorithm without having to tune more complex control parameters in applications, which makes it easily applicable to real-world optimization problems, such as mechanical design problems[7], automatic clustering problems[8] for multiple objectives, optimization design for a micro-channel heat sink[9], environment dispatch optimization in micro grids[10], PI controller optimization for photovoltaic distributed static compensator filter parameterization[11], tea category identification[12], surface grinding process optimization[13],

optimum power flow[14], solving combined economic emission dispatch solution[15], parameters selection for harmony search algorithm[16], and ball bearing design[17]. The basic Jaya algorithm was directly used to get better results than other algorithms in these applications.

However, there are deficiencies within the Jaya approach. In the Jaya algorithm, the optimal solution tends to approach the global best solution. In complex multi-dimensional and multi-modal optimization, there are cases where the variables of some dimensions near the optimal solution while the variables of the other dimensions are far from the optimal solution. In these cases, the solution is still considered as the global solution due to the low fitness, but it is easily led to trapped in the local optimum[18]. On the other hand, the optimal solution tends to move away from the worst solution. In the worst solution, there are also variables in some dimensions that are close to the optimal solution, but are considered to be avoided, which may result in a slow convergence.

A new mutation strategy was added to the basic Jaya, which was called improved Jaya algorithm (IJA), and found optimal solutions of operational costs[19] .

A novel enhanced Jaya algorithm with a two group adaption (E-Jaya) is proposed in this paper. E-Jaya algorithm includes two modifications. First, in E-Jaya, candidates in the population are separated into the better group and the worse group according to their fitness. The mean of the better group and the mean of the worse group are used in the updated equation. Thus, the E-Jaya tends to reach the mean of the better group and to avoid the mean of the worse group. Second, note, to have non algorithm-specific parameters in E-Jaya, a new adaptive method of dividing the two groups is developed.

The remainder of this paper is organized as follows: Section 2 describes the details of Jaya algorithm and E-Jaya. Section 3 lists twelve benchmark functions and initialization. Section 4 presents the simulation results, and Section 5 provides our conclusion.

2. Enhanced Jaya Algorithm with Group Strategy Adaption

2.1. Jaya Algorithm

Assume the objective function *f(x)* is minimized, and suppose that the number of design variables is m (*h=1, 2, . . . , m*), the total number of iteration is g (*j=1, 2, . . . , g*), and the population size is n (*i=1, 2, . . . , n*). Of all possible candidate solutions, suppose the best one is the best value of *f(x)* and the worst is the worst value of *f(x)*. If $X_{h,i,j}$ denotes the value of the h^{th} variable during the j^{th} iterations for the i^{th} candidate, this value is updated as follows[6]:

$$X'_{h,i,j} = X_{h,i,j} + r_{1,h,j}\left(X_{h,best,j} - \left|X_{h,i,j}\right|\right) - r_{2,h,j}\left(X_{h,worst,j} - \left|X_{h,i,j}\right|\right) \quad (1)$$

where $X_{h,best,j}$ is the value of variable h for the best value of *f(x)* and $X_{h,worst,j}$ is the value of variable h for the worst value of *f(x)*. $X'_{h,i,j}$ is a new value of $X_{h,i,j}$. $r_{1,h,j}$ and $r_{2,h,j}$ are two random numbers in the scope [0, 1] of the h^{th} variable during the j^{th} iteration. The term "$r_{1,h,j}\left(X_{h,best,j} - \left|X_{h,i,j}\right|\right)$" shows the result as it approaches the best solution; the term "$r_{2,h,j}\left(X_{h,worst,j} - \left|X_{h,i,j}\right|\right)$" indicates the result that prevents the worst solution. $X'_{h,i,j}$ is accepted if the better value of *f(x)* is given. All accepted values of variables are retained at the end of each iteration and are considered in terms of the input for the next iteration. r_1 and r_2 affect the exploration capacity in the global search space, while the term ($\left|X_{h,i,j}\right|$) in Eq. (1) affects the ability to exploit the local search space. Only general control parameters such as the population size *n* and the number of generations *g* are required in the Jaya algorithm, thus it is fairly convenient for use in practical

Algorithm 1. Jaya algorithm framework.

1: initialization: the population size n, the number of design variables m and the number of generations g
2: randomly select n candidate solutions
3: **repeat**
4: assess their fitness
5: identify Best and worst solutions
6: calculate $X'_{h,i,j}$ based on Equation (1)
7: **if** $X'_{h,i,j} < LB \vee X'_{h,i,j} > UB$ **then**
8: randomly select r from U(0,1)
9: $X'_{h,i,j} \leftarrow LB + r \cdot (UB - LB)$
10: **endif**
11: **if** $f(X'_{h,i,j}) < f(X_{h,i,j})$ **then**
12: $X_{h,i,j} = X'_{h,i,j}$
13: $f_{min} = f(X'_{h,i,j})$
14: **endif**
15: **until** stop condition is reached
16: **return** the minimum fitness f_{min} and the corresponding solution $X_{h,i,j}$

applications. where $x \in [LB, UB]$. U(a, b) denotes an uniform distribution between a and b, and r is a random number.

Algorithm 1 describes the full version of the Jaya algorithm.

2.2. Enhanced Jaya Algorithm with a Two Group Adaption

2.2.1 Enhanced Jaya Algorithm with Two Groups

Initially the population is ranked in ascending order from the best to the worst based on the fitness *f(x)* of each candidate solution, and then divided into two groups. Note, the better group G_b contains n_b elements and the worse group G_w of n_w elements. n_b and n_w satisfy the following equation.

$$n_b + n_w = n \tag{2}$$

where n is population size.

Then, the mean of the better group $X_{h,mbetter,j}$ and the mean of the worse group $X_{h,mworse,j}$ are calculated using the following equations:

$$X_{h,mbetter,j} = \frac{\sum_{i=1}^{n_b} X_{h,i,j}}{n_b} \tag{3}$$

$$X_{h,mworse,j} = \frac{\sum_{i=1}^{n_w} X_{h,i,j}}{n_w} \tag{4}$$

Finally, the enhanced Jaya algorithm with two groups (E-Jaya1) is updated as follows:

$$X'_{h,i,j} = X_{h,i,j} + r_{1,h,j}\left(X_{h,mbetter,j} - |X_{h,i,j}|\right) - r_{2,h,j}\left(X_{h,mworse,j} - |X_{h,i,j}|\right) \tag{5}$$

Algorithm 2 describes the main part of E-Jaya1.

Algorithm 2. Main part of E-Jaya1.

1: assess their fitness
2: sort the fitness in ascending order
3: choose n_b and n_w based on Equation (2)
4: calculate $X_{h,mbetter,j}$ and $X_{h,mworse,j}$ based on Equations (3) and (4)
5: calculate $X'_{h,i,j}$ based on Equation (5)

The complete E-Jaya1 can be obtained, if algorithm 2 is substituted for lines 4 through 6 of algorithm 1.

2.2.2 An adaptive method of dividing the two groups

The ratio of the better group r_b can be defined by the following:

$$r_b = \frac{n_b}{n} \tag{6}$$

where $0 < r_b < 1$. Provided that the parameter r_b is determined, the values of n_b and n_w can be calculated according to equations (6) and (2).

Based on our experiments, a reasonable r_b is within the range of [0.5, 0.9]. A notable characteristic of the Jaya algorithm is that it does not need to set any algorithm-specific control parameters, thus, to have non algorithm-specific parameters in E-Jaya is also very necessary, which means that the parameter r_b should be adaptive. A new adaptive method was developed to generate the parameter r_b. The specific approach of this method is to produce a uniformly distributed random number between 0.5 and 0.9 for the parameter r_b.

Algorithm 3 describes the adaptive method of dividing the two groups.

Algorithm 3. The adaptive method of dividing the two groups.

1: generate random numbers between 0.5 and 0.9 for r_b
2: calculate n_b based on Equation (6)

Algorithm 4. E-Jaya algorithm framework.

1: initialization: the population size n, the number of design variables m and the number of generations g
2: randomly select n candidate solutions
3: **repeat**
4: generate random numbers between 0.5 and 0.9 for r_b
5: calculate n_b based on Equation (6)
6: assess their fitness
7: sort the fitness in ascending order
8: choose n_b and n_w based on Equation (2)
9: calculate $X_{h,mbetter,j}$ and $X_{h,mworse,j}$ based on Equations (3) and (4)
10: calculate $X'_{h,i,j}$ based on Equation (5)
11: **if** $X'_{h,i,j} < LB \vee X'_{h,i,j} > UB$ **then**
12: randomly select r from U(0,1)
13: $X'_{h,i,j} \leftarrow LB + r \cdot (UB - LB)$
14: **endif**
15: **if** $f(X'_{h,i,j}) < f(X_{h,i,j})$ **then**
16: $X_{h,i,j} = X'_{h,i,j}$
17: $f_{min} = f(X'_{h,i,j})$
18: **endif**
19: **until** stop condition is reached
20: **return** the minimum fitness f_{min} and the corresponding solution $X_{h,i,j}$

2.2.3 Enhanced Jaya Algorithm with a Two Group Adaption

In E-Jaya, the adaptive method of generating the random number for the parameter r_b is aware of the adaptation. The sorted candidates $X_{h,i,j}$ ($i=1,2,\ldots,n$) are then divided into G_b and G_w according to their fitness, with n_b and n_w elements. Finally, the mean of the better group $X_{h,mbetter,j}$ and the mean of the worse group $X_{h,mworse,j}$ are used in Equation (5) for candidates updating.

Algorithm 4 describes the full version of the E-Jaya algorithm.

3. Benchmark Functions and Initialization

3.1. Benchmark Functions

To evaluate the performance of E-Jaya, twelve standardized benchmark functions[20] are employed. The functions are uni-modal and multi-modal. The global minimum is zero.

Table 1 presents the functions of uni-modal (F1~F8) and multi-modal (F9~F12) and their ranges.

3.2. Initialization

We ran the E-Jaya and other algorithms on over 50 independent runs with the same population (n=20), and the number of function evaluation (NFE) are also the same for different algorithms. Several statistical measures were used in order to fully assess the performance of E-Jaya as compared with the other algorithms, including the best, the worst, mean and median values, and standard deviations.

We ran the simulations in Matlab R2012b software on a laptop running 4GB memory, 2.6 GHZ CPU, and 64-bit Windows 7.

4. Simulation Studies and Discussions

4.1. Comparison with Jaya Algorithms

4.1.1 Enhanced Jaya with Two Groups

To evaluate the effectiveness of the E-Jaya1, the twelve functions were tested in different dimensions, such as 40, 60 and 100, for Jaya and E-Jaya1 with different parameters n_b.

Table 1. Twelve benchmark functions.

Function	Range
$F1(x)=\sum_{i=1}^{n} x_i^2$	[-100,100]
$F2(x)=\sum_{i=1}^{n} \lvert x_i\rvert+\prod_{i=1}^{n} \lvert x_i\rvert$	[-10.10]
$F3(x)=\sum_{i=1}^{n} ix_i^2$	[-5.12,5.12]
$F4(x)=10^6x_1^2+\sum_{i=2}^{n} x_i^2$	[-100,100]
$F5(x)=\sum_{i=1}^{n} (\mathrm{floor}(x_i+0.5))^2$	[-100,100]
$F6(x)=\sum_{i=1}^{n} x_i^2+\sum_{i=1}^{n} ix_i^4$	[-100,100]
$F7(x)=\sum_{i=1}^{n} (ix_i)^2$	[-100,100]
$F8(x)=\sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	[-10,10]
$F9(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	[-32,32]
$F10(x)=\sum_{i=1}^{n/4}((x_{4i-3}+10x_{4i-2})^2+5(x_{4i-1}-x_{4i})^2+(x_{4i-2}-2x_{4i-1})^4+10(x_{4i-3}-x_{4i})^4)$	[-10,10]
$F11(x)=0.1\left(sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2\left(1+sin^2(3\pi x_i+1)\right)(x_n-1)^2\left(1+sin^2(2\pi x_n)\right)\right)+\sum_{i=1}^{n} u(x_i,5,100,4)$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a\le x_i\le a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	[-50,50]
$F12(x)=\frac{\pi}{n}\left(10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2\left(1+10sin^2(\pi y_{i+1})\right)+(y_n-1)^2\right)+\sum_{i=1}^{n} u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}$	[-50,50]

Table 2. Comparison between Jaya and E-Jaya1 for different n_b with D=40.

Functions	Jaya	E-Jaya1						NFE
		n_b=8	n_b=10	n_b=12	n_b=14	n_b=16	n_b=18	
F1	5.48E-09	5.42E-15	2.81E-16	3.62E-17	8.74E-18	**3.48E-19**	1.15E-18	40000
F2	1.24E-07	5.11E-12	1.09E-08	2.01E-12	1.66E-14	**3.08E-16**	9.02E-18	60000
F3	5.03E-10	2.24E-16	2.07E-17	2.04E-12	2.99E-19	**4.50E-20**	8.16E-20	40000
F4	4.15E-05	1.72E-07	1.40E-10	3.16E-10	5.52E-12	**1.37E-12**	2.36E-20	30000
F5	2.25E+01	3.80E-01	3.00E-01	1.96E+00	2.00E-01	**2.20E-01**	1.20E+00	50000
F6	7.62E-06	1.26E-10	8.74E-12	4.72E-13	9.26E-14	**7.60E-15**	1.06E-14	40000
F7	2.99E-02	6.67E-03	9.20E-03	1.03E-02	1.32E-02	**7.71E-03**	8.03E-03	60000
F8	5.40E+02	2.63E+02	2.38E+02	2.36E+02	2.33E+02	**2.31E+02**	1.93E+02	40000
F9	5.09E+00	2.58E-14	2.21E-14	2.05E-14	1.99E-14	**2.24E-14**	3.97E-02	80000
F10	3.45E-04	6.26E-06	3.77E-06	3.05E-06	1.36E-06	**3.89E-06**	6.17E-06	40000
F11	8.79E+00	3.52E-02	1.12E-02	1.26E-02	1.05E-02	**1.05E-01**	3.81E-01	50000
F12	2.23E+01	4.09E+00	2.78E+00	2.70E+00	1.89E+00	**1.65E+00**	2.04E+00	50000

The mean errors for Jaya and E-Jaya1 with different n_b while D=40, are presented in Table 2.

Once the group strategy was employed in Jaya (Table 2), the performance of Java can be greatly enhanced, as seen at F1, F2, F3, F4, F6 and F9. F5, F7, F10, F11 and F12 were also improved. When n_b=16, the best performance was achieved. As a result, the E-Jaya1 algorithm can improve Jaya's performance.

To find why the group strategy is good for Jaya, the curves for different values of the best, the worst, the mbetter and the mworse were shown in Fig. 1. The function is F1 and n_b=18.

It can be seen from the Figure 1, the values of the best and the worst change dramatically for the initial 50 iterations, but the values of the mbetter and the mworse quickly moves to the optimum zero point in just less than 20 iterations. Thus, E-Jaya1 can speed up the convergence rate and increase the accuracy of the solution.

The novel method to employ the behaviors of the better swarms and the worse swarms in E-Jaya1, can improve the performance of the Jaya algorithm, since Jaya only considers the behaviors of the two individuals that are best individual and the worst individual.

The mean errors for Jaya and E-Jaya1 with different n_b while D=60, are presented in Table 3.

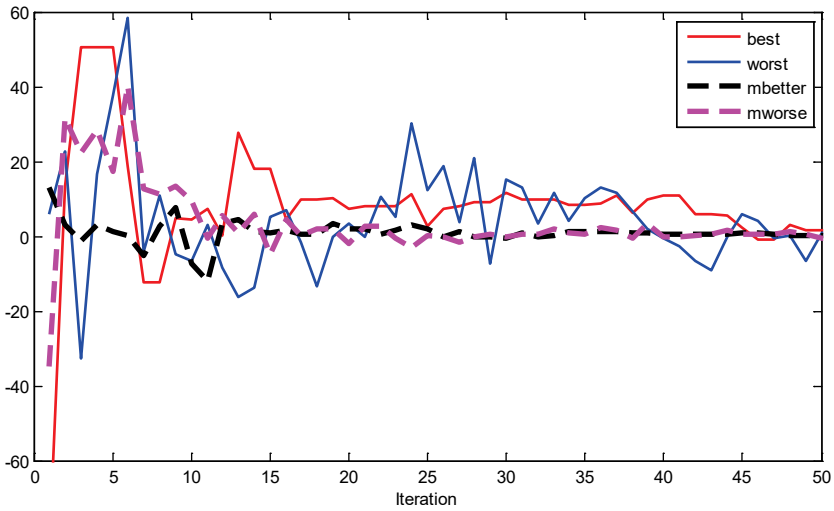

Fig. 1. The curves for values of the best, the worst, the mbetter and the mworse.

Table 3. Comparison between Jaya and E-Jaya1 for different n_b with D=60.

Functions	Jaya	E-Jaya1						NFE
		n_b=8	n_b=10	n_b=12	n_b=14	n_b=16	n_b=18	
F1	9.00E-06	4.28E-15	1.70E-15	**1.18E-16**	6.40E-18	1.97E-18	5.26E-17	60000
F2	2.57E-04	3.20E+00	3.23E+00	**1.14E-07**	1.63E-13	1.10E-15	5.38E-20	100000
F3	5.28E-06	2.12E-15	6.10E-17	**1.23E-17**	5.74E-19	1.21E-19	4.13E-18	60000
F4	8.87E-05	1.05E-13	2.90E-15	**1.51E-16**	2.46E-17	6.55E-18	2.40E-16	60000
F5	3.62E+02	1.16E+00	1.08E+00	**1.04E+00**	2.30E+00	1.64E+00	9.42E+00	80000
F6	2.85E-02	7.89E-08	6.10E-10	**2.03E-11**	1.59E-12	3.36E-13	2.16E-12	60000
F7	1.64E-01	2.43E-02	2.06E-02	**3.63E-02**	4.41E-02	8.18E-02	3.61E-02	60000
F8	1.30E+03	6.93E+02	6.55E+02	**6.27E+02**	6.22E+02	6.42E+02	5.49E+02	60000
F9	6.90E+00	2.88E-13	4.19E-14	**3.55E-14**	4.37E-02	2.10E-02	6.44E-01	100000
F10	1.21E-02	2.42E-05	2.77E-05	**1.33E-06**	5.68E-06	3.34E-06	9.96E-06	60000
F11	3.63E+01	1.45E+01	1.41E+00	**6.73E-01**	2.41E-01	6.74E-01	3.61E+00	80000
F12	3.62E+01	2.51E+01	9.56E+00	**7.84E+00**	6.06E+00	5.07E+00	8.11E+00	80000

Table 4. Comparison of Jaya and E-Jaya1 for different n_b with D=100.

Functions	Jaya	E-Jaya1						NFE
		n_b=8	n_b=10	n_b=12	n_b=14	n_b=16	n_b=18	
F1	3.71E-02	4.55E-19	2.92E-22	1.12E-24	**1.60E-26**	9.50E-27	1.63E-21	150000
F2	5.65E-01	3.44E+01	4.50E+00	1.95E+01	**4.23E-19**	1.96E-28	1.35E-19	250000
F3	1.46E-03	3.69E-20	2.06E-23	2.77E-25	**4.18E-27**	4.07E-27	7.87E-23	150000
F4	8.21E-01	3.98E-19	2.10E-22	3.30E-24	**1.61E-25**	5.49E-26	2.48E-21	150000
F5	3.04E+03	6.14E+00	8.08E+00	5.14E+00	**1.07E+01**	1.58E+01	7.28E+01	300000
F6	1.54E-01	9.52E-09	2.00E-13	1.81E-17	**2.54E-19**	4.98E-20	2.06E-15	150000
F7	1.32E-01	3.93E-02	2.29E-02	2.78E-02	**2.38E-02**	3.27E-02	4.96E-02	150000
F8	3.32E+03	1.89E+03	1.73E+03	1.63E+03	**1.63E+03**	1.61E+03	1.39E+03	150000
F9	1.29E+01	4.10E-02	2.05E-02	4.22E-02	**1.72E-01**	6.37E-01	2.33E+00	300000
F10	1.99E+01	1.67E-06	2.25E-08	1.99E-08	**4.17E-09**	3.78E-05	4.62E-02	200000
F11	2.49E+01	3.91E+02	1.81E+01	7.00E+00	**5.64E+00**	9.59E+00	3.16E+01	250000
F12	4.91E+01	1.42E+03	4.10E+01	2.07E+01	**1.42E+01**	1.20E+01	1.59E+01	250000

The results from the Table 3 indicate that the E-Jaya1 clearly increases the performance of Jaya for all functions. The best performance was obtained when n_b=12.

The mean errors for Jaya and E-Jaya1 with different n_b while D=100, are presented in Table 4.

The results from the Table 4 show that even in the high dimensions of 100, the E-Jaya1 can still significantly enhance the performance of Jaya. The accuracy of the mean solution was improved from 10^{-2} to 10^{-26} for F1, from 10^{-1} to 10^{-19} for F2, from 10^{-3} to 10^{-27} for F3, from 10^{-1} to 10^{-25} for F4 , from 10^{-1} to 10^{-19} for F6 , from 10^{1} to 10^{-1} for F9 ,and from 10^{1} to 10^{-9} for F10. The best performance was obtained when n_b=14.

The results in the Table 2, Table 3 and Table 4 indicate that the E-Jaya1 significantly increases the performance of Jaya. The best performance of E-Jaya1 for different dimensions will be chosen for later comparison. The reasonable parameter r_b is between 0.5 and 0.9.

4.1.2 Enhanced Jaya Algorithm with a Two Groups Adaption

To validate the adaptive method by dividing the two groups effectively, E-Jaya was tested on the twelve functions with dimensions of 40, 60 and 100. Comparisons among Jaya, E-Jaya1 and E-Jaya were also conducted.

The statistical results with D=40 for Jaya, E-Jaya1 and E-Jaya are presented in Table 5.

The statistical results indicate that all values of E-Jaya1 are much better than that of Jaya. Using the adaptive method of dividing the two groups, E-Jaya obtains the similar results to E-Jaya1, with the exception of F7 and F10. The best results of E-Jaya for F7 and

Table 5. Comparison results with D=40 for Jaya, E-Jaya1 and E-Java.

Fun.	Term	Jaya	E-Jaya1	E-Jaya
F1	Best	7.10E-11	1.15E-20	**4.11E-21**
	Mean	5.48E-09	3.48E-19	**2.79E-19**
	Std.Dev	7.17E-09	**5.98E-19**	6.37E-19
F2	Best	1.25E-10	6.23E-19	**1.38E-19**
	Mean	1.24E-07	3.08E-16	**6.64E-17**
	Std.Dev	8.33E-07	2.03E-15	**3.74E-16**
F3	Best	3.13E-12	**1.57E-22**	1.95E-22
	Mean	5.03E-10	4.50E-20	**1.72E-20**
	Std.Dev	2.26E-09	1.33E-19	**4.49E-20**
F4	Best	1.05E-06	2.26E-14	**1.77E-14**
	Mean	4.15E-05	**1.37E-12**	1.99E-12
	Std.Dev	1.07E-04	**2.44E-12**	6.23E-12
F5	Best	3.00E+00	0	**0**
	Mean	2.25E+01	**2.20E-01**	4.00E-01
	Std.Dev	3.75E+01	**4.18E-01**	5.71E-01
F6	Best	1.67E-07	1.45E-17	**2.60E-18**
	Mean	7.62E-06	7.60E-15	**2.57E-15**
	Std.Dev	1.43E-05	2.72E-14	**5.15E-15**
F7	Best	8.29E-07	**1.65E-07**	1.10E-05
	Mean	2.99E-02	7.71E-03	**7.13E-03**
	Std.Dev	9.23E-02	1.22E-02	**1.16E-02**
F8	Best	3.52E+02	1.22E+02	**9.35E+01**
	Mean	5.40E+02	2.31E+02	**2.04E+02**
	Std.Dev	9.97E+01	6.89E+01	**5.66E+01**
F9	Best	1.42E+00	1.51E-14	**7.99E-15**
	Mean	5.09E+00	2.24E-14	**2.06E-14**
	Std.Dev	5.63E+00	5.65E-15	**4.77E-15**
F10	Best	3.39E-07	**3.75E-10**	2.72E-09
	Mean	3.45E-04	**3.89E-06**	8.07E-06
	Std.Dev	5.69E-04	**1.23E-05**	3.30E-05
F11	Best	8.68E-14	**1.93E-20**	6.21E-20
	Mean	8.79E+00	1.05E-01	**8.91E-03**
	Std.Dev	7.35E+00	4.69E-01	**1.52E-02**
F12	Best	8.02E+00	3.48E-01	**3.11E-01**
	Mean	2.23E+01	1.65E+00	**1.28E+00**
	Std.Dev	6.33E+00	1.85E+00	**1.30E+00**

Table 6. Comparison results with D=60 for Jaya, E-Jaya1 and E-Java.

Fun.	Term	Jaya	E-Jaya1	E-Jaya
F1	Best	1.28E-08	1.13E-18	**2.01E-20**
	Mean	9.00E-06	1.18E-16	**1.37E-18**
	Std.Dev	2.11E-05	2.87E-16	**2.63E-18**
F2	Best	9.98E-08	2.91E-20	**7.26E-23**
	Mean	2.57E-04	1.14E-07	**4.04E-21**
	Std.Dev	1.03E-03	5.82E-07	**9.78E-21**
F3	Best	1.26E-08	9.35E-20	**8.56E-22**
	Mean	5.28E-06	1.23E-17	**6.24E-20**
	Std.Dev	2.59E-05	4.18E-17	**9.26E-20**
F4	Best	8.97E-08	2.85E-18	**1.56E-20**
	Mean	8.87E-05	1.51E-16	**6.25E-18**
	Std.Dev	2.11E-04	2.30E-16	**1.31E-17**
F5	Best	4.10E+01	0	**0**
	Mean	3.62E+02	**1.04E+00**	1.92E+00
	Std.Dev	5.55E+02	2.39E+00	**2.35E+00**
F6	Best	2.03E-04	8.04E-14	**4.06E-16**
	Mean	2.85E-02	2.03E-11	**4.76E-13**
	Std.Dev	6.48E-02	3.48E-11	**2.04E-12**
F7	Best	**3.64E-06**	1.16E-05	1.14E-04
	Mean	1.64E-01	3.63E-02	**2.44E-02**
	Std.Dev	3.41E-01	1.26E-01	**3.38E-02**
F8	Best	6.73E+02	4.74E+02	**3.55E+02**
	Mean	1.30E+03	6.27E+02	**5.75E+02**
	Std.Dev	2.25E+02	**8.85E+01**	1.10E+02
F9	Best	2.96E+00	**2.22E-14**	2.58E-14
	Mean	6.90E+00	**3.55E-14**	7.45E-02
	Std.Dev	3.37E+00	**5.27E-15**	2.55E-01
F10	Best	1.05E-04	2.86E-12	**1.71E-12**
	Mean	1.21E-02	**1.33E-06**	2.35E-06
	Std.Dev	2.31E-02	**2.38E-06**	9.19E-06
F11	Best	2.06E-02	3.78E-16	**9.90E-20**
	Mean	3.63E+01	**6.73E-01**	7.55E-01
	Std.Dev	1.73E+01	**2.08E+00**	2.47E+00
F12	Best	1.69E+01	1.41E+00	**8.81E-01**
	Mean	3.62E+01	7.84E+00	**5.08E+00**
	Std.Dev	8.69E+00	4.48E+00	**3.13E+00**

F10 are slightly worse than that of E-Jaya1, but the mean results are similar. Thus, the adaptive method of dividing the two groups is effective.

The statistical results with D=60 for Jaya, E-Jaya1 and E-Jaya are presented in Table 6.

The results from Table 6 show that all statistical values of E-Jaya1 are also much better than that of Jaya, with the exception of the best value for F7. By using the adaptive method of dividing the two groups, E-Jaya obtains about the same results compared with E-Jaya1, with the exception of F9, F1, F2, F3, F4 and F6. The mean result of E-Jaya for F9 is much worse than that of E-Jaya1, but still outperforms Jaya. The results of E-Jaya for F1, F2, F3, F4 and F6 are superior to that of E-Jaya1. Thus, E-Jaya can significantly enhance the performance of Jaya.

The statistical results with D=100 for Jaya, E-Jaya1 and E-Jaya are also presented in Table 7.

The results from Table 7 also show that E-Jaya1 significantly improves the performance of statistical values for Jaya. By using the adaptive method of dividing the two groups, E-Jaya also achieves comparable results to that of E-Jaya1, with the exception of F7, F10 and F11. The best results of E-Jaya for F7 and F11 are worse than that of E-Jaya1, and the mean result of E-Jaya for F10 is worse than that of E-Jaya1, but they are still better than that of Jaya. The results of E-Jaya for F2, F3, F4 and F6 are superior to that of E-Jaya1. Thus, E-Jaya can significantly enhance the performance of Jaya.

The results from Table 5, Table 6 and Table 7 indicate that the E-Jaya can replace the E-Jaya1. And

Table 7. Comparison results with D=100 for Jaya, E-Jaya1 and E-Java.

Fun.	Term	Jaya	E-Jaya1	E-Jaya
F1	Best	4.23E-06	2.10E-28	**2.45E-29**
	Mean	3.71E-02	**1.60E-26**	2.97E-26
	Std.Dev	2.28E-01	**2.29E-26**	1.09E-25
F2	Best	4.89E-05	1.68E-32	**2.09E-34**
	Mean	5.65E-01	4.23E-19	**1.95E-30**
	Std.Dev	1.63E+00	2.99E-18	**9.09E-30**
F3	Best	8.93E-07	1.01E-29	**9.29E-30**
	Mean	1.46E-03	4.18E-27	**2.71E-27**
	Std.Dev	6.39E-03	1.06E-26	**9.08E-27**
F4	Best	2.36E-05	4.77E-28	**3.06E-28**
	Mean	8.21E-01	1.61E-25	**6.15E-26**
	Std.Dev	5.75E+00	3.61E-25	**1.34E-25**
F5	Best	1.14E+03	**0**	1.00E+00
	Mean	3.04E+03	**1.07E+01**	3.52E+01
	Std.Dev	1.57E+03	**2.55E+01**	1.15E+02
F6	Best	1.32E-03	6.43E-22	**5.18E-23**
	Mean	1.54E-01	2.54E-19	**1.19E-20**
	Std.Dev	3.17E-01	6.39E-19	**2.41E-20**
F7	Best	1.01E-05	**7.76E-06**	1.62E-05
	Mean	1.32E-01	2.38E-02	**1.54E-02**
	Std.Dev	2.27E-01	**3.99E-02**	2.47E-01
F8	Best	2.44E+03	1.23E+03	**1.15E+03**
	Mean	3.32E+03	1.63E+03	**1.57E+03**
	Std.Dev	4.48E+02	**2.10E+02**	2.19E+02
F9	Best	9.78E+00	**4.35E-14**	4.71E-14
	Mean	1.29E+01	**1.72E-01**	5.33E-01
	Std.Dev	1.94E+00	**4.01E-01**	5.47E-01
F10	Best	3.66E-04	5.60E-25	**2.19E-30**
	Mean	1.99E+01	**4.17E-09**	2.56E-06
	Std.Dev	8.62E+01	**2.38E-08**	1.79E-05
F11	Best	4.11E-12	**7.67E-30**	1.19E-27
	Mean	2.49E+01	**5.64E+00**	8.11E+00
	Std.Dev	4.25E+01	**8.66E+00**	1.15E+01
F12	Best	2.96E+01	5.24E+00	**2.24E+00**
	Mean	4.91E+01	1.42E+01	**1.20E+01**
	Std.Dev	2.01E+01	5.48E+00	**4.83E+00**

the adaptive method by dividing the two groups is effective.

statistical results were shown in Table 5 for dimensions of 40, Table 6 for dimensions of 60 and Table 7 for dimensions of 100. The results indicate that the E-Jaya algorithm can significantly enhance the performance of the Jaya algorithm.

The comparison curves of the mean values between Jaya and E-Jaya with different dimensions of 40, 60, and 100 are shown in Figure 2.

Note, the blue line represents the Jaya curve, and the red line represents the E-Jaya curve. The solid line is use for curves with dimensions of 40, the dashed line is used for curves with dimensions of 60, and the dotted line is used for curves with dimensions of 100.

Fig. 2 shows that the E-Jaya algorithm is superior to the Jaya algorithm in terms of the solution accuracy.

4.2. Comparison with Other Swarm Intelligence Algorithms

4.2.1. Mean Errors Comparison

Additionally, E-Jaya was compared with other swarm intelligent algorithms, including a differential evolution (DE), a self-adapting control parameters in differential evolution[21] (jDE), a FA, and a SPSO2011[22]. The NFE and population size are the same for each function. The parameters for these algorithms are shown in Table 8.

Setting the parameters of these algorithms is a challenge, taken from Ref. 21 for DE and jDE, and Ref.22 for SPSO2011. For FA, the reduction of randomness can accelerate the convergence rate. $\alpha^{t+1} = (1-\delta)a^t$ in Ref.23 is used to decrease α gradually. δ is a small constant associated with the iteration.

Table 8. Parameters settings.

Algorithms	Parameters
DE	F=0.5, CR=0.9, popSize=40
jDE	$\tau_1 = \tau_2 = 0.1$, F ∈ [0.1, 1.0], CR ∈ [0, 1]
FA	$\alpha = 0.9$, $\beta = 0.25$, $\gamma = 1.0$
SPSO2011	w = 0.7213, c1 = 1.1931, c2 = 1.1931

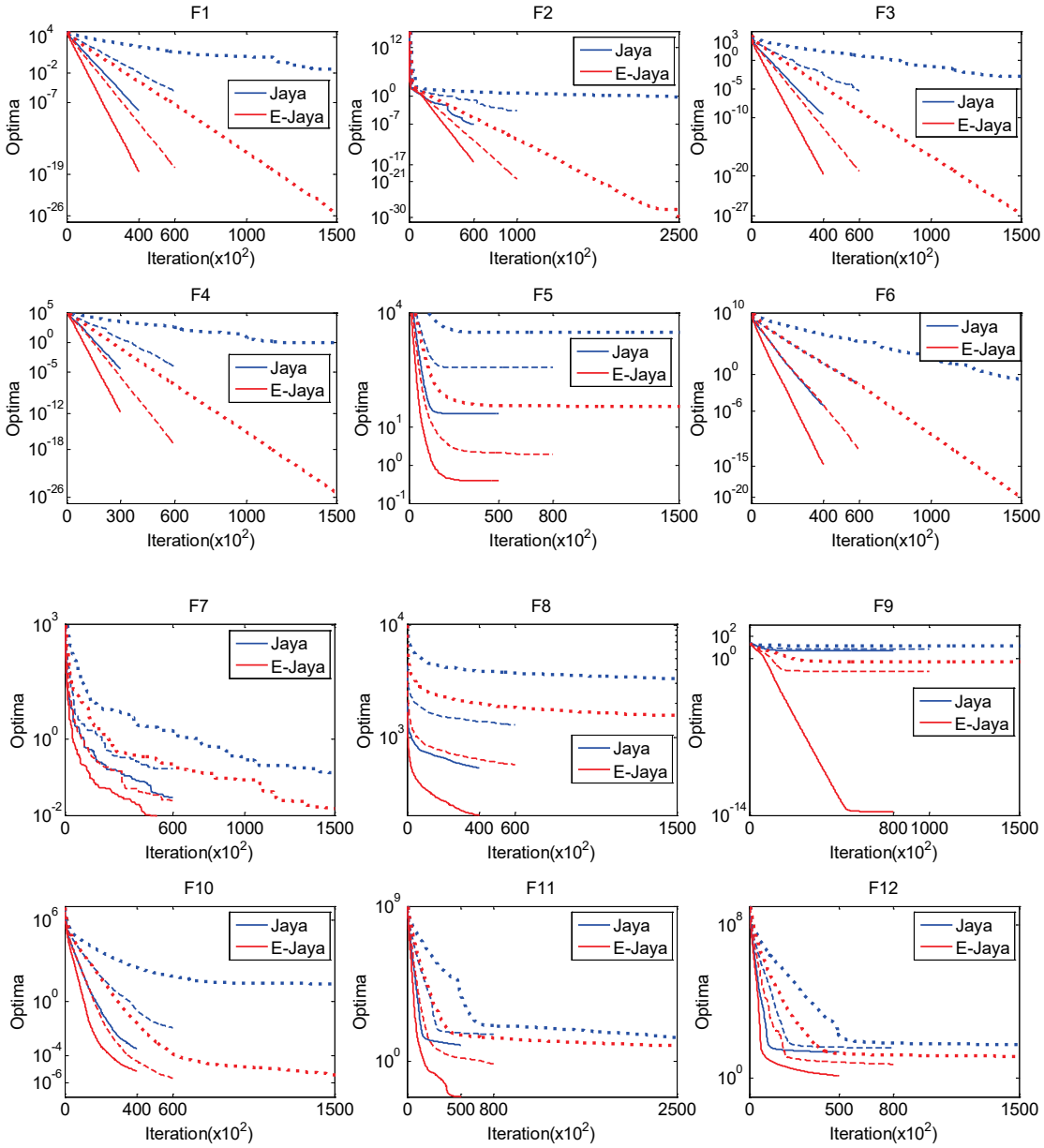

Fig. 2. The comparison curves between Jaya and E-Jaya with different dimensions

Table 9. Mean errors for the five algorithms with D=40.

Fun.	DE	jDE	FA	SPSO2011	E-Jaya	NFE
F1	2.19E-01	6.92E-19	1.43E-03	2.42E-13	2.79E-19	40000
F2	1.63E-07	2.27E-20	9.77E-02	5.47E+00	6.64E-17	60000
F3	3.83E-04	1.82E-20	2.75E-01	3.37E-01	1.72E-20	40000
F4	1.89E-01	7.07E-13	1.48E-02	1.69E+03	1.99E-12	30000
F5	1.11E+01	9.70E+00	4.00E-01	1.21E+01	4.00E-01	50000
F6	1.12E+05	1.06E-15	1.53E-03	1.41E-13	2.57E-15	40000
F7	9.31E-03	2.74E-03	2.62E+05	2.78E+02	7.13E-03	60000
F8	3.71E+00	1.40E+01	3.01E+02	3.26E+02	2.04E+02	40000
F9	1.08E+00	1.47E-01	2.48E-02	2.08E+01	2.06E-14	80000
F10	7.54E-02	4.32E-09	2.58E-01	3.50E-02	8.07E-06	40000
F11	3.46E+04	1.70E-02	3.44E-04	5.17E+01	8.91E-03	50000
F12	1.16E+02	2.80E-02	2.29E+01	1.87E-02	1.28E+00	50000

Table 10. Ranks for the five algorithms when D= 40

Fun.	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	Rank
DE	5	3	3	4	4	5	3	1	4	4	5	5	3.83
jDE	2	1	2	1	3	1	1	2	3	1	3	2	1.83
FA	4	4	4	3	1	4	5	4	2	5	1	4	3.41
SPSO2011	3	5	5	5	5	3	4	5	5	3	4	1	4.00
E-Jaya	1	2	1	2	1	2	2	3	1	2	2	3	1.83

Table 11. Mean errors for the five algorithms with D=60.

Fun.	DE	jDE	FA	SPSO2011	E-Jaya	NFE
F1	1.86E+00	5.27E-17	3.51E-03	3.95E-13	1.37E-18	60000
F2	1.34E-06	8.58E-22	2.41E-01	1.43E+01	4.04E-21	100000
F3	6.80E-03	2.15E-18	9.21E-01	1.37E+00	6.24E-20	60000
F4	7.10E-01	1.41E-15	1.11E-02	2.15E+03	6.25E-18	60000
F5	1.00E+02	4.40E+01	6.80E-01	5.42E+01	1.92E+00	80000
F6	1.43E+05	6.40E-11	3.74E-03	6.29E-13	4.76E-13	60000
F7	4.46E-02	3.33E-02	2.43E+06	1.61E+04	2.44E-02	60000
F8	1.90E+01	5.68E+01	4.65E+02	2.43E+02	5.75E+02	60000
F9	2.69E+00	2.24E+00	8.25E-03	2.09E+01	7.45E-02	100000
F10	7.54E-01	1.87E-08	3.41E-01	4.38E-02	2.35E-06	60000
F11	4.43E+05	2.52E-01	6.68E-04	1.36E+02	7.55E-01	80000
F12	9.98E+03	6.64E-02	2.73E+01	2.80E-02	5.08E+00	80000

Tables 9 presents the mean errors for the five algorithms with D=40.

Ranks for the five algorithms when D=40 is shown in Table 10, based on the results from Table 9.

The results from Table 10 show that E-Jaya and jDE both rank higher (1.83) among the five algorithms.

The mean errors for the five algorithms with D=60 is presented in Table 11.

Ranks for the five algorithms when D=60 is shown in Table 18, according to the results from Table 11.

The results in Table 12 indicate that E-Jaya algorithm ranks the highest (2.00) among the five algorithms.

The mean errors for the five algorithms with D=100 is described in Tables 13.

According to the data in Table 13, Table 14 presents the ranks for the five algorithms when D=100.

The results in Table 14 also indicate that E-Jaya ranks the highest (1.92) among the five algorithms, based on the solution accuracy.

4.2.2 Runtime Cost Comparison

The runtime cost for twelve function with D=40 is compared to the above algorithms, see Fig. 3.

The results in Figure 3 indicate that the time cost of E-Jaya is the least and runtime cost of the jDE is the third most expensive among five algorithms. For solution accuracy, E-Jaya and jDE rank similarly, but E-Jaya has lower runtime costs when compared to jDE. Thus, E-Jaya is better than jDE in terms of solution accuracy and runtime cost.

Fig. 4 shows the runtime cost of the five algorithms for twelve function with D=60.

The results from Fig. 4 show that the time cost of E-Jaya ranks the fourth most expensive, and jDE ranks the second most expensive among the five algorithms. The time cost of FA is the least, but the difference of time cost between FA and E-Jaya is extremely small.

Table 12. Ranks for the five algorithms when D= 60

Fun.	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	Rank
DE	5	3	3	4	5	5	3	1	4	5	5	5	4.00
jDE	2	1	2	2	3	3	2	2	3	1	2	2	2.08
FA	4	4	4	3	1	4	5	4	1	4	1	4	3.25
SPSO2011	3	5	5	5	4	2	4	3	5	3	4	1	3.67
E-Jaya	1	2	1	1	2	1	1	5	2	2	3	3	2.00

Table 13. Mean errors for the five algorithms with D=100.

Fun.	DE	jDE	FA	SPSO2011	E-Jaya	NFE
F1	1.26E+01	2.14E-20	7.19E-03	1.17E-19	2.97E-26	150000
F2	1.14E-07	2.18E-27	2.86E-01	3.19E+01	1.95E-30	250000
F3	1.11E+00	1.66E-20	2.29E+00	2.48E+00	2.71E-27	150000
F4	1.05E+01	5.41E-21	1.65E-02	2.72E+03	6.15E-26	150000
F5	8.79E+02	3.54E+02	1.98E+00	2.29E+02	3.52E+01	300000
F6	4.05E+06	1.07E-15	7.69E-03	1.33E-19	1.19E-20	150000
F7	4.36E-02	5.16E-02	1.90E+07	6.77E+04	1.54E-02	150000
F8	5.06E+01	1.31E+02	1.84E+03	1.55E+03	1.57E+03	150000
F9	6.64E+00	6.01E+00	2.93E-02	2.02E+01	5.33E-01	300000
F10	3.47E+01	3.66E-14	2.12E-01	1.45E-02	2.56E-06	200000
F11	7.92E+05	4.34E-01	1.63E-03	2.55E+02	8.11E+00	250000
F12	3.48E+05	1.19E-01	3.00E+00	1.49E-02	1.20E+01	250000

Table 14. Ranks for the five algorithms when D= 100

Fun.	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	Rank
DE	5	3	3	4	5	5	2	1	4	5	5	5	3.91
jDE	2	2	2	2	4	3	3	2	3	1	2	2	2.33
FA	4	4	4	3	1	4	5	5	1	4	1	3	3.25
SPSO2011	3	5	5	5	3	2	4	3	5	3	4	1	3.58
E-Jaya	1	1	1	1	2	1	1	4	2	2	3	4	1.92

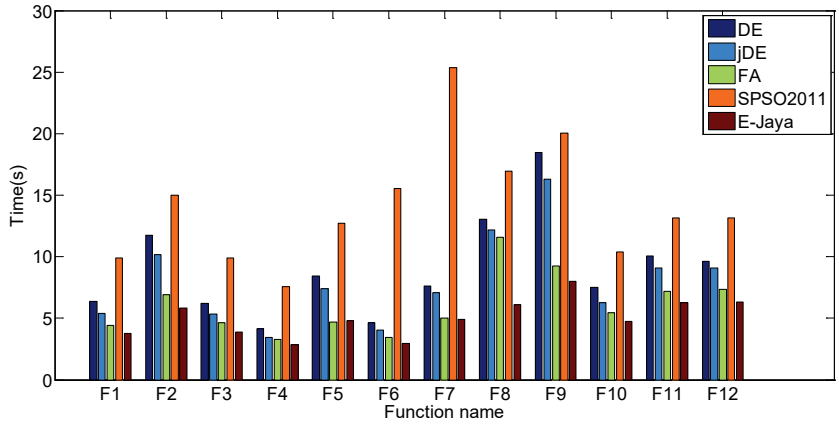

Fig. 3. Comparison for runtime cost of the five algorithms with D=40.

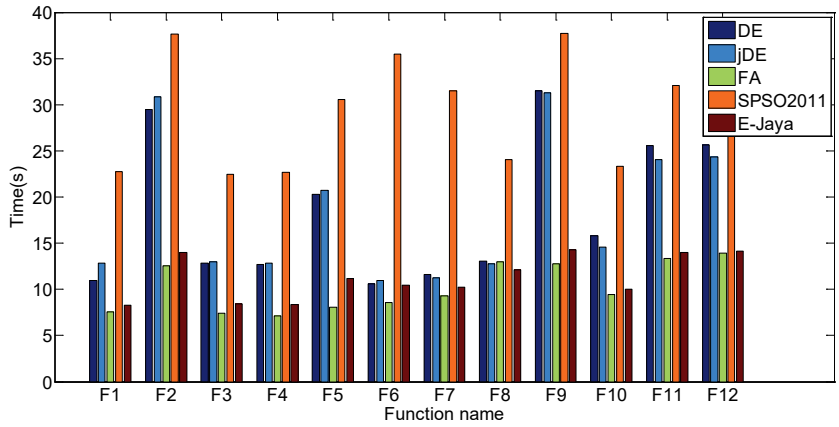

Fig. 4. Comparison for runtime cost of the five algorithms with D=60.

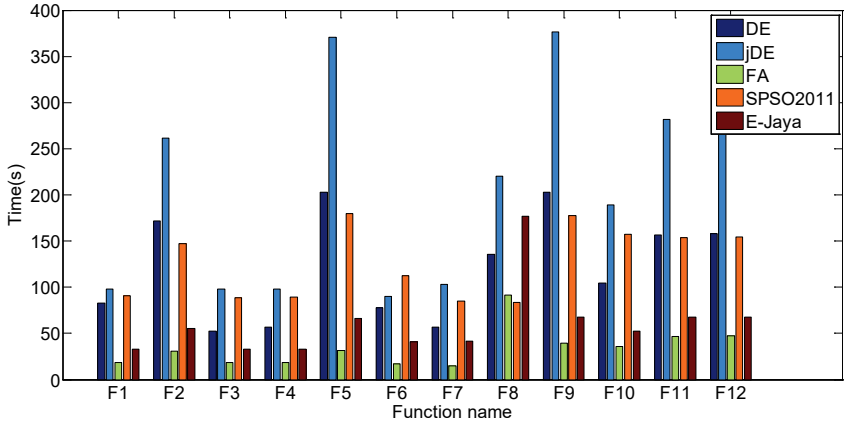

Fig. 5. Comparison for runtime cost of the five algorithms with D=100.

For example, the runtime cost of FA and E-Jaya are 13.93s and 14.11s for F12, respectively. Thus, they are considered to be the similar.

Fig. 5 depicts the runtime cost of the five algorithms for twelve function with D=100.

The results from Fig. 5 also indicate that the time cost of jDE is the highest, and the time cost of FA and E-Jaya are the least and the second less, respectively. Thus, for the solution accuracy, the E-Jaya algorithm ranks the highest, and for the runtime cost, it ranks the second highest.

Overall, the E-Jaya algorithm ranks the highest according to the comparison of the solution accuracy and the runtime cost of the five algorithms among the different dimensions.

5. Conclusions

A novel E-Jaya algorithm was proposed in this paper to enhance the performance of the Jaya algorithm. In E-Jaya, the mean of the better group and the mean of the worse group are used to find an optimal solution, where E-Jaya performed significant better than Jaya in terms of the solution accuracy.

The new adaptive method of dividing the two groups is also effective. Thus, E-Jaya does not need to set any algorithm-specific parameter. The E-Jaya algorithm ranks the highest when compared with DE, jDE, FA, and SPSO2011.

Instead of considering the behaviors of only the two individuals (the best and the worst), in E-Jaya, the behaviors of all swarms, which are the better swarms and the worse swarms, are considered. Only twelve benchmark functions with different dimensionality are assessed using the E-Java algorithm in this paper. The E-Java algorithm will be validated in the practical problems. In addition, this novel method of group strategy will be further studied and possibly used in our future work for other algorithms.

Acknowledgements

The author is thankful to the anonymous reviewers for their valuable comments to improve the technical content and the presentation of the paper.

References

1. M. Dorigo, Learning and Natural Algorithms, PhD. Thesis, Politecnicodi Milano, Milan, Italy, 1992.
2. J. Kennedy and R.C. Eberhart, Particles warm optimization, in *Proc. IEEE Int. Conf. Neural Networks*, (Piscataway, NJ, USA, 1995), pp. 1942–1948.
3. R. Storn and K.Price, Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces, *J. Glob. Optim*. **11**(1997) 341–359.
4. X. S. Yang, Firefly algorithms for multimodal optimization, in *Proc. 5th Int. Conf. Stochastic Algorithms: Foundation and Applications*, (Sapporo, Japan, 2009), pp. 169–17
5. X. S. Yang, A new metaheuristic bat-inspired algorithm, in *Nature Inspired Cooperative Strategies for Optimization (NISCO 2010)* (Springer: Berlin, Germany, 2010), pp. 65–74.
6. R. V. Rao, Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems, *Int. J. Ind. Eng. Comput*, **7** (2016)19–34.
7. R. V. Rao and G. G. Waghmare, A new optimization algorithm for solving complex constrained design optimization problems, *Eng. Optimiz*, **48** (2016) 1–24.
8. R. R. Kurada1 and K. P. Kanadam, Automatic unsupervised data classification using Jaya evolutionary algorithm, *Adv. Comput. Intell.: Int. J*, **3** (2016) 35–42.
9. R. V. Rao, K. C. More, J, Taler and P. Ocłon, Dimensional optimization of a micro-channel heat sink using Jaya algorithm, *Appl. Therm. Eng,* **103** (2016) 572–582.
10. I. N. Trivedi, S. N. Purohit, P. Jangir and M. T. Bhoye, Environment dispatch of distributed energy resources in a micro grid using Jaya algorithm, in *Proc. Int. Conf. Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB16)*, (Chennai, India, 2016), pp. 224–228.
11. S. Mishra and P. K. Ray, Power quality improvement using photovoltaic fed DSTATCOM based on Jaya optimization, *IEEE Trans. Sust. Energy*, **99** (2016) 1–9.
12. Y. D. Zhang, X. J. Yang, C. Cattani, R. V. Rao, S. H. Wang and P. Phillips, Tea category identification using a novel fractional Fourier entropy and Jaya algorithm, *Entropy*, **18** (2106), 77.
13. R. V. Rao, D. P. Rai and J. Balic, Surface grinding process optimization using Jaya algorithm, in *Computational Intelligence in Data Mining—Volume 2, Advances in Intelligent Systems and Computing* (Springer: India, 2015), pp. 487–495.
14. W. Warid, H. Hizam, N. Mariun and N. I. Abdul-Wahab, Optimal power flow using the Jaya algorithm, *Energies,* **9** (2016) 678.
15. S. Phulambrikar, Solving combined economic emission dispatch solution using Jaya optimization algorithm approach, *Int. Res. J. Eng. Tech*, **3** (2016) 501-512
16. A. Alaa, A. Rahman and A. Alsewari, Hyperdize Jaya algorithm for harmony search algorithm's parameters

selection, in *Proc. National Conf. Postgraduate Research (NCON-PGR 2016),* (University Malaysia Pahang, Pekan, 2016) pp. 788-791.
17. A. Kumar, Ball bearing design through Jaya algorithm, in *Proc. 5th Int. Conf. recent trends in engineering, science & management* (Pune, India, 2016), pp.1304-1333.
18. J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, *IEEE Trans. Evolut. Comput*, **10** (2006), 281–295.
19. A. R. Azizipanah, F. Golestaneh and H.B. Gooi, Corrective economic dispatch and operational cycles for probabilistic unit commitment with demand response and high wind power, *Appl. Energy* **182** (2016), 634–651.
20. M. Mitic and Z. Miljkovic, Chaotic fruit fly optimization algorithm, *Knowl. Based Syst*, **89** (2015), 446–458.
21. J. Brest, S. Greiner, B. Boskovic, M. Mernik and V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, *IEEE Trans. Evolut. Comput*, **6** (2006), 646–657.
22. M. Zambrano, M. Bigiarini and R. Rojas, Standard particle swarm optimization 2011 at CEC-2013: A baseline for future PSO improvements, in *Proc. 2013 IEEE Congress on Evolutionary Computation (CEC)*, (Cancun, Mexico, 2013), pp. 2337–2344.
23. X. S. Yang, *Nature-Inspired Metaheuristic Algorithms,* 2nd ed. (Luniver Press, Frome, UK, 2010).