Earthworm optimization algorithm: a bio-inspired metaheuristic algorithm for global optimization problems

Article *in* International Journal of Bio-Inspired Computation · October 2015

DOI: 10.1504/IJBIC.2015.10004283

CITATIONS
95

READS
3,598

3 authors:

Gai-Ge Wang
Ocean University of China
124 PUBLICATIONS **4,350** CITATIONS

SEE PROFILE

Suash Deb
CVRCE, India
120 PUBLICATIONS **8,679** CITATIONS

SEE PROFILE

Leandro Coelho
Pontifícia Universidade Católica do Paraná (PUC-PR)
426 PUBLICATIONS **7,206** CITATIONS

SEE PROFILE

Earthworm optimisation algorithm: a bio-inspired metaheuristic algorithm for global optimisation problems

Gai-Ge Wang*

School of Computer Science and Technology,
Jiangsu Normal University,
Xuzhou, Jiangsu, 221116, China
and
Institute of Algorithm and Big Data Analysis,
Northeast Normal University,
Changchun, 130117, China
and
School of Computer Science and Information Technology,
Northeast Normal University,
Changchun, 130117, China
Email: gaigewang@163.com
Email: gaigewang@gmail.com
*Corresponding author

Suash Deb

Cambridge Institute of Technology,
Cambridge Village, Tatisilwai,
Ranchi 835103, Jharkhand, India
Email: suashdeb@gmail.com

Leandro dos Santos Coelho

Industrial and Systems Engineering Graduate Program (PPGEPS),
Pontifical Catholic University of Parana (PUCPR),
Curitiba, Parana, Brazil
and
Electrical Engineering Graduate Program (PPGEE),
Department of Electrical Engineering,
Polytechnic Center, Federal University of Parana (UFPR),
Curitiba, Parana, Brazil
Email: lscoelho2009@gmail.com

Abstract: Earthworms can aerate the soil with their burrowing action and enrich the soil with their waste nutrients. Inspired by the earthworm contribution in nature, a new kind of bio-inspired metaheuristic algorithm, called earthworm optimisation algorithm (EWA), is proposed in this paper. The EWA method is inspired by the two kinds of reproduction (Reproduction 1 and Reproduction 2) of the earthworms. Reproduction 1 generates only one offspring by itself. Reproduction 2 is to generate one or more than one offspring at one time, and this can successfully be done by nine improved crossover operators. In addition, Cauchy mutation (CM) is added to EWA method. Nine different EWA methods with one, two and three offsprings based on nine improved crossover operators are respectively proposed. The results show that EWA23 performs the best and it can find the better fitness on most benchmarks than others.

Keywords: earthworm optimisation algorithm; EWA; evolutionary computation; benchmark functions; improved crossover operator; Cauchy mutation; CM; bio-inspired metaheuristic; global optimisation; swarm intelligence; evolutionary algorithms.

Reference to this paper should be made as follows: Wang, G-G., Deb, S. and Coelho, L.d.S. (xxxx) 'Earthworm optimisation algorithm: a bio-inspired metaheuristic algorithm for global optimisation problems', *Int. J. Bio-Inspired Computation*, Vol. X, No. Y, pp.xxx–xxx.

Biographical notes: Gai-Ge Wang obtained his Bachelor in Computer Science and Technology from Yili Normal University, Yining, China, in 2007. His PhD degree was in the field of computational intelligence and its applications at the Chinese Academy of Sciences. He is

currently an Associate Professor in the School of Computer Science and Technology at Jiangsu Normal University, Xuzhou, China. He has published over 50 journal papers and conference papers, and more than 30 papers are indexed by SCI/EI. Furthermore, he is referee of other 50 international journals (Elsevier, IEEE and Springer). He is a member of IEEE, the International Society for Metaheuristic Optimization in Science and Technology (ISMOST), Publications Chairs of ISCMI 2015, and International Program Committee Member in more than 30 conferences. His research interests are swarm intelligence, soft computing, evolutionary computation and its applications in engineering, such as scheduling and path planning.

Suash Deb specialises in metaheuristics and has published extensively. He has been the recipient of the Albert Einstein International Award for Scientific Excellence, Rajiv Gandhi Education Excellence Award, etc. He was awarded UNDP Fellowship for visiting Stanford University, USA, CIMPA-INRIA-UNESCO Fellowship, ICTP Fellowship, etc. He was the Asian Expert of ARPA, Department of Defense, Federal Government of USA. Currently belonging to CIT, he served institutions like NCKBC, NIST, CVRCE, etc. He is the Editor-in-Chief of the *International Journal of Soft Computing and Bioinformatics*, Regional Editor of the *Neural Computer and Applications*, etc. He was elected as the Founding President of the INNS-India Regional Chapter and also the Secretary of the IEEE CIS, Calcutta Chapter. He is the General Chair of ISCBI15 & ISCMI15, the flagship event and 5th commemorative event of INNS-India respectively. He is a senior member of IEEE.

Leandro dos Santos Coelho received his BS in Computer Science and BS in Electrical Engineering from the Federal University of Santa Maria, Brazil, in 1994 and 2000, respectively. After, he earned his MS and Doctoral in Computer Science and Electrical Engineering from the Federal University of Santa Catarina, in 1997 and 2000, respectively. He is currently a full Professor at the Department of Automation and Systems, Pontifical Catholic University of Parana and Associate Professor at the Department of the Electrical Engineering, Federal University of Parana, Brazil. He is interested in natural computing, optimisation methods, machine learning and control systems.

1 Introduction

With the development of real-life engineering techniques, calculation amount significantly increases as real-life engineering problems are becoming more and more complex. Traditional methods based on formal logics or mathematical programming have difficulty in solving such complicated problems fast and efficiently. In order to cope with this limitation, a variety of modern nature-inspired intelligent algorithms have been put forward and applied to solve optimisation problems. Among various nature-inspired algorithms, swarm-based algorithms and evolutionary algorithms (EAs) are two of the most representative paradigms among them.

Swarm-based algorithms, so called swarm intelligence methods, are one of the most well-known paradigms in nature-inspired algorithms which have been widely used in various applications. The inspiration of the swarm intelligence algorithms is collective behaviour of animals. Two of widely used swarm intelligence (Yang and Cui, 2014) are particle swarm optimisation (PSO) (Kennedy and Eberhart, 1995; Wang et al., 2014c) and ant colony optimisation (ACO) (Dorigo et al., 1996). The idea of PSO (Kennedy and Eberhart, 1995) originated from the social behaviour of bird flocking when searching for the food. The ants in nature are well capable of keeping the past paths in mind by pheromone. Inspired by this phenomenon, the ACO algorithm is proposed by Dorigo et al. (1996). Recently, more excellent swarm intelligence algorithms have been proposed, such as artificial bee colony (ABC) (Karaboga and Basturk, 2007; Li and Yin, 2012), animal migration optimisation (AMO) (Li et al., 2014) cuckoo search (CS) (Li et al., 2013; Srivastava et al., 2012; Wang et al., 2014d; Yang and Deb, 2009), artificial plant optimisation algorithm (APOA) (Cui et al., 2013a, 2013b), monarch butterfly optimisation (MBO) (Wang et al., 2015), bat algorithm (BA) (Cai et al., 2014; Mirjalili et al., 2013; Xue et al., 2015; Yang, 2010), ant lion optimiser (ALO) (Mirjalili, 2015a), wolf search algorithm (WSA) (Fong et al., 2015), multi-verse optimiser (MVO) (Mirjalili et al., 2015), dragonfly algorithm (DA) (Mirjalili, 2015b), and krill herd (KH) (Gandomi and Alavi, 2012; Gandomi et al., 2013). They are inspired by the smarm behaviour of honey bees, animals, cuckoos, butterflies, bats, wolves and krill, respectively.

By simplifying and idealising the genetic evolution process, different kinds of EAs have been proposed and used in a wide range of applications. Genetic algorithm (GA) (Goldberg, 1998), evolutionary programming (EP) (Bäck, 1996), genetic programming (GP) (Hand, 1992) and evolutionary strategy (ES) (Beyer and Schwefel, 2002) are four of the most classical EAs among them. With the development of the evolutionary theory, some new methods have been proposed over the last decades that significantly improved the theory and search capacities of EAs. Differential evolution (DE) (Hu et al., 2011; Storn and Price, 1997) is a very efficient search algorithm that simulates the biological mechanisms of natural selection and mutation. On the other hand, DE uses a rather greedy and less stochastic approach than classical EAs such GAs. The best-to-survive criterion is adopted in the above algorithms on a population of solutions. Stud genetic algorithm (SGA) (Khatib and Fleming, 1998; Wang et al., 2014b) is a special kind of GA that uses the best individual

and the other randomly selected individuals at each generation for crossover operator. By incorporating the sole effect of predictor variable as well as the interactions between the variables into the GP, Gandomi and Alavi (2011) proposed an improved version of GP algorithm, called multi-stage genetic programming (MSGP), for nonlinear system modelling. Recently, motivated by the natural biogeography, Simon has provided the mathematics of biogeography and accordingly proposed a new kind of EA: biogeography-based optimisation (BBO) (Li and Yin, 2013; Mirjalili et al., 2014; Saremi et al., 2014; Simon, 2008; Wang et al., 2014a).

A new kind of bio-inspired metaheuristic algorithm, called earthworm optimisation algorithm (EWA), is proposed in this paper. The EWA method is inspired by the two kinds of reproduction of the earthworms in nature. In EWA, the offspring are generated through Reproduction 1 and Reproduction 2 independently, and then, we used weighted summation of all the generated offsprings to get the final earthworm for next generation. Reproduction 1 generates only one offspring by itself that is also a special kind of reproduction in nature. Reproduction 1 is to generate one or more than one offspring at one time, and this can successfully done by several improved crossover operators that are an extended version of classical crossover operator used in DE and GA. In order to make certain earthworms escape from local optima and improve the search ability of earthworms, the addition of a Cauchy mutation (CM) is made to the EWA method. This can also help the whole earthworm individuals proceed to the better positions. For the purpose of showing the optimisation process of EWA method, nine different EWA methods with one, two and three offsprings based on nine improved crossover operators are respectively proposed and they are compared between each other through 22 high-dimensional benchmarks. The results show that EWA23 performs the best and is further compared with seven state-of-art metaheuristic algorithms on 48 benchmark functions and an engineer case. The EWA method is able to find the better function values on most benchmark problems than seven other metaheuristic algorithms.

The goals of this paper are three-fold. Firstly, we want to give a general presentation of the new optimisation method called EWA. We do this by first studying the two kinds of reproduction, and then generalising it to formulate a general-purpose metaheuristic method. Secondly, we want to compare EWA with other population-based optimisation methods. We do this by looking at the commonalities and differences from an algorithmic point-of-view, and also by comparing their performances on an array of benchmark functions. Thirdly, EWA is used to solve the real-world problem of test-sheet composition (TSC). This will prove the availability of EWA to real-world engineering problems.

The remainder of this paper is organised as follows. Section 2 reviews the reproduction and regeneration behaviour of earthworms in nature, and Section 3 discusses how the reproduction behaviour of earthworms can be used to formulate a general-purpose search heuristic. Nine different improved crossover operators that are the extended version of the classical crossover operator are also proposed by simulating Reproduction 2 of the earthworms in this section. With the aim of the showing the performance of EWA method, several simulation results comparing EWA with themselves and other optimisation methods, both for general benchmark functions and for an engineering case, are presented in Section 4. Section 5 presents some concluding remarks and suggestions for further work.

2 Reproduction and regeneration behaviour of earthworms

An earthworm is a kind of tube-shaped, segmented animal generally found in soil that absorbs nutrients from live and dead organic matter. It has a very strong digestive system that runs through the whole body. It breathes through its skin. An earthworm has a central and a peripheral nervous system. Its moves depend on muscles that are located on the outer boundary of each segment (Hickman et al., 2006).

2.1 Reproduction

The reproduction of earthworms occurs most frequently at night. Earthworms have both male and female sexual organs (generally called hermaphrodites). The locations of the sexual organs are in segments 9 to 15 with one or two pairs of testes contained within sacs. The sperm expelled from segment 15 is generated, saved and ejected by the two or four pairs of seminal vesicles through the male pores. The eggs are generated by ovaries in segment 13 through female pores on segment 14. The sperm from the other worm during copulation are obtained by internal sacs in segments 9 and 10. Therefore, for each worm, the sperms are transferred from segment 15 to segments 9 and 10. Some species use external spermatophores for sperm transfer. The above information may be different depending on the species.

2.2 Regeneration

Earthworms are well capable of regenerating lost segments, but different species have different regeneration ability. After studying regeneration in different kinds of species for 20 years, Gates published some of his findings that show for certain species it has possibility of growing into two intact ones from a bisected specimen. The main points of Gates's reports are given as follows:

- Eisenia fetida with head regeneration, in an anterior direction, possible at each intersegmental level back to and including 23/24, while tails were regenerated at any levels behind 20/21, i.e., two worms may grow from one (Gates, 1949).
- Perionyx excavatus readily regenerated lost parts of the body, in an anterior direction from as far back as 17/18, and in a posterior direction as far forward as 20/21.

- Criodrilus lacuum also has prodigious regenerative capacity with 'head' regeneration from as far back as 40/41 (Gates, 1953).

An unidentified Tasmanian earthworm shown growing a second head has been reported (Blakemore, 2001).

In this paper, the reproduction and regeneration of the earthworms are considered as two kinds of reproduction of earthworms, and then two kinds of reproduction are idealised to form a general-purpose optimisation method.

3 Earthworm optimisation algorithm

In order to make the reproduction behaviour of earthworms address various optimisation problems, the reproduction behaviour of earthworms can be idealised into the following rules.

1 All the earthworms in the population have the ability of reproducing the offsprings, and each earthworm individual has two and only two kinds of reproduction.

2 Each child earthworm individual generated by either of kinds contains all the genes whose length is equal to the parent earthworm.

3 The earthworm individuals with the best fitness pass on directly next generation, and they cannot be changed by any operators. This can guarantee that the earthworm population cannot deteriorate with the increment of generations.

In the following, the description of the two kinds of reproduction of earthworms is given.

3.1 The first kind of reproduction (Reproduction 1)

As discussed in the previous section, earthworms are hermaphrodites, that is, each individual carries both male and female sex organs. Therefore, the child earthworm can be generated by only one parent for certain earthworm. This reproduction process can be formulated as:

$$x_{i1,j} = x_{\max,j} + x_{\min,j} - \alpha x_{i,j} \quad (1)$$

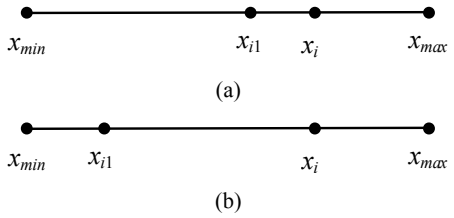
where $x_{i,j}$ indicates the j^{th} element of x_i that presents the position of the earthworm i. Similarly, $x_{i1,j}$ indicates the j^{th} element of x_{i1} that is the newly-generated position of the earthworm $i1$. $x_{\max,j}$ and $x_{\min,j}$ are respectively upper and lower bound of the position of earthworm i. $\alpha \in [0, 1]$ is a similarity factor that can determine the distance between the parent and child earthworm. If α is small, the distance between them is short, and x_{i1} is close to x_i. This makes the method implement the local search around the earthworm i [see Figure 1(a)]. When $\alpha = 0$, equation (1) turns into equation (2). The earthworm i and the newly-generated earthworm i has the biggest distance that is $x_{\max,j} + x_{\min,j}$.

$$x_{i1,j} = x_{\max,j} + x_{\min,j} \quad (2)$$

On the contrary, if α is big, the distance between them is long, and x_{i1} is far away from x_i. This encourages exploration and makes the method implement the global search with the long step [see Figure 1(b)]. When α is equal to 1, the equation (1) is becoming equation (3). The earthworm i and the newly-generated earthworm i has the smallest distance. At this point, the positions of the earthworm i and the newly-generated earthworm i are symmetric with regard to $(x_{\max,j} + x_{\min,j}) / 2$. In fact, equation (3) is essentially an oppositional-based learning method (Wang et al., 2011) that has been widely used in various stochastic optimisation methods. The mentioned equation is given by

$$x_{i1,j} = x_{\max,j} + x_{\min,j} - x_{i,j} \quad (3)$$

Figure 1 A simplified representation of Reproduction 1 with (a) small α and (b) big α



Through the above analyses, it can be seen that, the EWA method can balance the exploration and exploitation by adjusting similarity factor α. In the current work, for EWA method, the implementation of the global search mainly depends on Reproduction 1. Therefore, the relatively bigger α is used in this paper.

3.2 The second kind of reproduction (Reproduction 2)

Certain earthworms are able to generate more than one individual at one time, which is a special kind of reproduction for earthworms. Reproduction 2 can be generalised to propose an improved version of crossover operator that can be used to deal with optimisation problems. In this paper, M is the number of child earthworms that are used to generate the final new earthworm for Reproduction 2 is set to M. In theory, M can be any integers not less than 0, while in fact, M is 1, 2, and 3 in most cases. N is the number of parent earthworms; N must be integers that are more than 1. In the following, the description of the improved crossovers can be given for the following three cases. Note that, we use S, M, and U short for 'single point crossover', 'multipoint crossover', and 'uniform crossover', respectively. S1 indicates that only one offspring is generated by using single point crossover. The rest, such as S2, S3, M1, M2, M3, U1, U2, and U3, can be deduced by analogy.

3.2.1 $N = 2$, $M = 1$

For this case, two parent earthworms are selected by roulette wheel selection, and we can call them P_1 and P_2, respectively. The offsprings can be formed by P_1 and P_2, and it can be given as:

$$\mathrm{P}=\begin{bmatrix}\mathrm{P}_1\\ \mathrm{P}_2\end{bmatrix}. \tag{4}$$

3.2.1.1 Single point crossover (S1)

For the purpose of implementing the crossover operator, two temporary variables should be calculated firstly.

$$y_1=\mathrm{SD}\left(\mathrm{P}_1(r:D),\mathrm{P}_2(r:D)\right) \tag{5}$$

$$y_2=\mathrm{SD}\left(\mathrm{P}_2(r:D),\mathrm{P}_1(r:D)\right) \tag{6}$$

where SD indicates set difference of two arrays, and SD(**A**, **B**) returns the data in **A** that is not in **B**. **A** and **B** are two matrixes or vectors. $\mathrm{P}_1(r{:}D)$ and $\mathrm{P}_2(r{:}D)$ indicate the elements form r to D in P_1, and P_2, respectively. r is a randomly selected integer between 1 and D. D is the length of an earthworm individual, that is, it is the function dimension. Subsequently, the two offsprings are generated by the following equations.

$$x_{12}=\left[\mathrm{P}_2\left(1:D-|y_1|\right),y_1\right] \tag{7}$$

$$x_{22}=\left[\mathrm{P}_1\left(1:D-|y_2|\right),y_2\right] \tag{8}$$

where $|y_1|$ and $|y_2|$ are the length of y_1 and y_2, respectively.

$$x_{i2}=\begin{cases}x_{12} & rand<0.5\\ x_{22} & \text{else}\end{cases} \tag{9}$$

where *rand* is a random number that can be drawn certain distribution.

3.2.1.2 Multipoint crossover (M1)

For multipoint crossover, firstly, two random integer numbers r_1 and r_2 are generated. Let us suppose that the r_1 is always less than r_2. The two offsprings can be generated by the following equations.

$$x_{12}=\left[\mathrm{P}_1\left(1:r_1\right),\mathrm{P}_2\left(r_1+1:r_2\right),\mathrm{P}_1\left(r_2+1:D\right)\right] \tag{10}$$

$$x_{22}=\left[\mathrm{P}_2\left(1:r_1\right),\mathrm{P}_1\left(r_1+1:r_2\right),\mathrm{P}_2\left(r_2+1:D\right)\right] \tag{11}$$

The generated earthworm for Reproduction 2 can be determined by equation (9).

3.2.1.3 Uniform crossover (U1)

For uniform crossover, each element in the two offsprings x_{12} and x_{22} can be generated as equation (12) when a random number *rand* is bigger than 0.5.

$$\begin{aligned}x_{12,j}&=\mathrm{P}_{1,j}\\ x_{22,j}&=\mathrm{P}_{2,j}\end{aligned} \tag{12}$$

where $x_{12,j}$ and $x_{22,j}$ are the j^{th} element of the offsprings x_{12} and x_{22}. Similarly, $\mathrm{P}_{1,j}$ and $\mathrm{P}_{2,j}$ are the j^{th} element of the two parents P_1 and P_2.

Otherwise, they are updated as equation (13).

$$\begin{aligned}x_{12,j}&=\mathrm{P}_{2,j}\\ x_{22,j}&=\mathrm{P}_{1,j}\end{aligned} \tag{13}$$

And then, the generated earthworm x_{i2} for Reproduction 2 can be determined by equation (9).

In sum, the two offsprings x_{12} and x_{22} generated by uniform crossover with $M = 1$ can be formed as shown in Algorithm 1.

Algorithm 1 Uniform crossover with $M = 1$

Begin
  for $j = 1$ to D (all the elements in an earthworm individual)
  do
    if *rand* > 0.5 **then**
      Generate the j^{th} element of the offsprings x_{12} and x_{22} as equation (12).
    else
      Generate the j^{th} element of the offsprings x_{12} and x_{22} as equation (13).
    end if
  end for j
  Determine the generated earthworm x_{i2} for Reproduction 2 by equation (9).
End.

3.2.2 N = 2, M = 2

For this case, two parent earthworms P_1 and P_2 are selected by roulette wheel selection as equation (4).

3.2.2.1 Single point crossover (S2)

For the purpose of implementing the crossover operator, y_1 and y_2 are respectively computed as equation (5) and equation (6). And then, the two offsprings x_{12} and x_{22} are generated by equation (7) and equation (8), respectively. The newly-generated earthworm can be calculated as

$$x_{i2}=\omega_1 x_{12}+\omega_2 x_{22} \tag{14}$$

where ω_1 and ω_2 are weighting factors, and they can be computed as

$$\begin{cases}\omega_1=\dfrac{E_2}{E_1+E_2}\\ \omega_2=\dfrac{E_1}{E_1+E_2}\end{cases} \tag{15}$$

where E_1 and E_2 are fitness of the earthworm x_{12} and x_{22}, respectively.

3.2.2.2 Multipoint crossover (M2)

Here, the two offsprings x_{12} and x_{22} are generated by equation (10) and equation (11), respectively. The newly-generated earthworm can be calculated as equation (14).

3.2.2.3 Uniform crossover (U2)

For uniform crossover, the two offsprings x_{12} and x_{22} are generated by equation (12) or equation (13), respectively. And their corresponding weighting factors ω_1 and ω_2 are computed as equation (15) according to their new fitness. The newly-generated earthworm for Reproduction 2 can be calculated as equation (14).

Note that, if one of the two selected parent earthworms is the best earthworm that is called stud in SGA (Khatib and Fleming, 1998), Reproduction 2 in EWA method is essentially an SGA process.

3.2.3 $N = 3$, $M = 3$

For this case, similarly, three parent earthworms are selected by roulette wheel selection, and we can call them P_1, P_2 and P_3, respectively. The offsprings can be formed by three parents, and it can be given as

$$\mathrm{P} = \begin{bmatrix} \mathrm{P}_1 \\ \mathrm{P}_2 \\ \mathrm{P}_3 \end{bmatrix} \tag{16}$$

3.2.3.1 Single point crossover (S3)

For the purpose of implementing the crossover operator, the following temporary variables should be calculated firstly.

$$y_1 = \mathrm{SD}\left(\mathrm{P}_1(r:D), \mathrm{P}_3(r:D)\right) \tag{17}$$

$$y_2 = \mathrm{SD}\left(\mathrm{P}_3(r:D), \mathrm{P}_2(r:D)\right) \tag{18}$$

$$y_3 = \mathrm{SD}\left(\mathrm{P}_2(r:D), \mathrm{P}_1(r:D)\right) \tag{19}$$

Subsequently, the three offsprings are generated as

$$x_{31} = \left[\mathrm{P}_3\left(1:D-|y_1|\right), y_1\right] \tag{20}$$

$$x_{32} = \left[\mathrm{P}_2\left(1:D-|y_2|\right), y_2\right] \tag{21}$$

$$x_{33} = \left[\mathrm{P}_1\left(1:D-|y_3|\right), y_3\right] \tag{22}$$

where $|y_1|$, $|y_2|$ and $|y_3|$ are the length of y_1, y_2 and y_3, respectively.

3.2.3.2 Multipoint crossover (M3)

For multipoint crossover, similarly, two integer numbers r_1 and r_2 are randomly generated and $r_1 < r_2$. The three offsprings can be generated by the following equations.

$$x_{31} = \left[\mathrm{P}_1\left(1:r_1\right), \mathrm{P}_2\left(r_1+1:r_2\right), \mathrm{P}_3\left(r_2+1:D\right)\right] \tag{23}$$

$$x_{32} = \left[\mathrm{P}_2\left(1:r_1\right), \mathrm{P}_3\left(r_1+1:r_2\right), \mathrm{P}_1\left(r_2+1:D\right)\right] \tag{24}$$

$$x_{33} = \left[\mathrm{P}_3\left(1:r_1\right), \mathrm{P}_1\left(r_1+1:r_2\right), \mathrm{P}_2\left(r_2+1:D\right)\right] \tag{25}$$

3.2.3.3 Uniform crossover (U3)

For uniform crossover, each element in the three offsprings x_{31}, x_{32} and x_{33} can be generated as equation (26) when a random number *rand* is bigger than 0.5.

$$\begin{aligned} x_{31,j} &= \mathrm{P}_{2,j} \\ x_{32,j} &= \mathrm{P}_{3,j} \\ x_{33,j} &= \mathrm{P}_{1,j} \end{aligned} \tag{26}$$

where $x_{31,j}$, $x_{32,j}$ and $x_{33,j}$ are the j^{th} element of the offsprings x_{31}, x_{32} and x_{33}, respectively. Similarly, P*1*,j, $\mathrm{P}_{2,j}$ and $\mathrm{P}_{3,j}$ are the j^{th} element of the three parents P_1, P_2 and P_3.

Otherwise, they are updated as equation (27).

$$\begin{aligned} x_{31,j} &= \mathrm{P}_{3,j} \\ x_{32,j} &= \mathrm{P}_{1,j} \\ x_{33,j} &= \mathrm{P}_{2,j} \end{aligned} \tag{27}$$

In sum, the three offsprings x_{31}, x_{32} and x_{33} generated by uniform crossover with $M = 3$ can be formed as shown in Algorithm 2.

After generating the offsprings x_{31}, x_{32} and x_{33}, the new earthworm for Reproduction 2 is generated as

$$\begin{aligned} x_{i2} &= \sum_{j=1}^{3} \omega_j x_{3j} \\ &= \omega_1 x_{31} + \omega_2 x_{32} + \omega_3 x_{33} \end{aligned} \tag{28}$$

where ω_1, ω_2 and ω_3 are weighting factors, and they can be computed as

$$\begin{cases} \omega_1 = \dfrac{1}{2}\dfrac{E_2 + E_3}{E_1 + E_2 + E_3} \\ \omega_2 = \dfrac{1}{2}\dfrac{E_1 + E_3}{E_1 + E_2 + E_3} \\ \omega_3 = \dfrac{1}{2}\dfrac{E_1 + E_2}{E_1 + E_2 + E_3} \end{cases} \tag{29}$$

where E_1, E_2 and E_3 are fitness of the earthworms x_{31}, x_{32} and x_{33}, respectively.

Algorithm 2 Uniform crossover with $M = 3$

Begin
  for j = 1 to D (all the elements in an earthworm individual)
  do
    if *rand* > 0.5 **then**
      Generate the j^{th} element of the offsprings x_{31}, x_{32} and x_{33} as equation (26).
    else
      Generate the j^{th} element of the offsprings x_{31}, x^{32} and x_{33} as equation (27).
    end if
  end for j
  Determine the generated earthworm x_{i2} for Reproduction 2 by equation (9).
End.

According to the above-mentioned cases when M = 1, 2 and 3, the computation of x_{i2} can be generalised based on M offsprings.

$$x_{i2} = \sum_{j=1}^{M} \omega_j x_{Mj} \tag{30}$$

where the weighting factor ω_j can be calculated as

$$\begin{aligned} \omega_j &= \frac{1}{M-1} \frac{\sum_{k=1,k\neq j}^{M} E_k}{\sum_{k=1}^{M} E_k} \\ &= \frac{1}{M-1} \frac{\begin{array}{c} E_1 + E_2 + \cdots + E_{j-1} \\ +E_{j+1} + \cdots + E_{M-1} + E_M \end{array}}{\begin{array}{c} E_1 + E_2 + \cdots + E_{j-1} + E_j \\ +E_{j+1} + \cdots + E_{M-1} + E_M \end{array}} \end{aligned} \tag{31}$$

where E_k is the fitness of the k^{th} offspring.

After the implementation of the two kinds of the reproduction, the position of the earthworm i for the next generation can be formed as

$$x_i' = \beta x_{i1} + (1-\beta) x_{i2} \tag{32}$$

where β is the proportional factor that can adjust the proportion of the x_{i1} and x_{i2} produced by the above kinds of reproduction. It can be given as

$$\beta^{t+1} = \gamma \beta^t \tag{33}$$

where t is current generation. Not that, the initial β is set to 1 when t = 0, that is β_0 = 1. γ is a constant that is similar to cooling factor of a cooling schedule in the simulated annealing (Kirkpatrick et al., 1983).

It can be seen from equation (33), at first, the proportional factor β is relatively big, and the bigger β makes the EWA method implement global search due to the x_{i1}. The proportional factor β decreases with the increment of generations. This indicates that the proportion of the x_{i1} produced by Reproduction 1 becomes less and less with the increment of generations. That is, the x_{i2} produced by Reproduction 2 becomes more and more with the increment of generations. This leads to the EWA method searching locally mainly at the end of search process. From the above analyses, we can draw the conclusion that, like similarity factor α, the proportional factor β is also able to adjust the balance of the global search and local search efficiently.

3.3 Cauchy mutation

For the purpose of escaping from local optima and improving the search ability of earthworms, the addition of a CM is made to the EWA method. This can also help the whole earthworm individuals proceed to the better positions. The CM has been successfully combined with many metaheuristic algorithms, such as PSO (Wu, 2011), DE (Ali and Pant, 2010). The 1D Cauchy density function is defined by:

$$f(x) = \frac{1}{\pi} \frac{t}{\tau^2 + x^2}, x \in R \tag{34}$$

where $\tau > 0$ is a scale parameter (Wang et al., 2007). The Cauchy distributed function is

$$F_\tau(x) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{x}{\tau}\right) \tag{35}$$

This mutation operator is well capable of decreasing the possibility of trapping into a local optimum. The description of the CM operator in EWA is given below:

$$W_j = \left(\sum_{i=1}^{NP} x_{i,j}\right) \Big/ NP \tag{36}$$

where $x_{i,j}$ is the j^{th} position vector of the i^{th} earthworm, NP is the population size, and W_j is a weight vector.

$$x_{i,j}' = x_{i,j} + W_j * C \tag{37}$$

where C is a random number drawn from a Cauchy distribution with τ = 1.

Algorithm 3 Earthworm optimisation algorithm

Begin

Step 1: **Initialisation**. Set the generation counter t = 1; initialise the population P of NP earthworm individuals randomly with uniform distribution in the search space; set the number of the kept earthworms $nKEW$, the maximum generation $MaxGen$, the similarity factor α, the initial proportional factor β_0, and the constant γ = 0.9 when computing the proportional factor β.

Step 2: **Fitness evaluation**. Evaluate each earthworm individual according to its position.

Step 3: **While** the best solution is not found or $t < MaxGen$ **do**

Sort all the earthworm individuals according to their fitness.

for i = 1 to NP (for all earthworm individuals) **do**

// Implement Reproduction 1.

Generate x_{i1} though Reproduction 1 as Section 3.1;

// Implement Reproduction 2.

// This process is essentially the implementation of an improved crossover operator.

if $i > nKEW$ **then**

Define the number of selected parents (N) and the generated offsprings (M);

Select N parents by using roulette wheel selection;

Generate M offsprings as Sections 3.2.1–3.2.3;

Computing x_{i2} according to M generated offsprings by equation (30);

else

Randomly select an earthworm individual as x_{i2}.

end if

Update the position of the earthworm i by equation (32).

end for i

for $j = nKEW + 1$ to NP (for all non-kept earthworm individuals) **do**

Implement Cauchy mutation as Section 3.3;

end for j

Evaluate the population according to the newly updated positions;

$t = t + 1$.

Step 4: **end while**

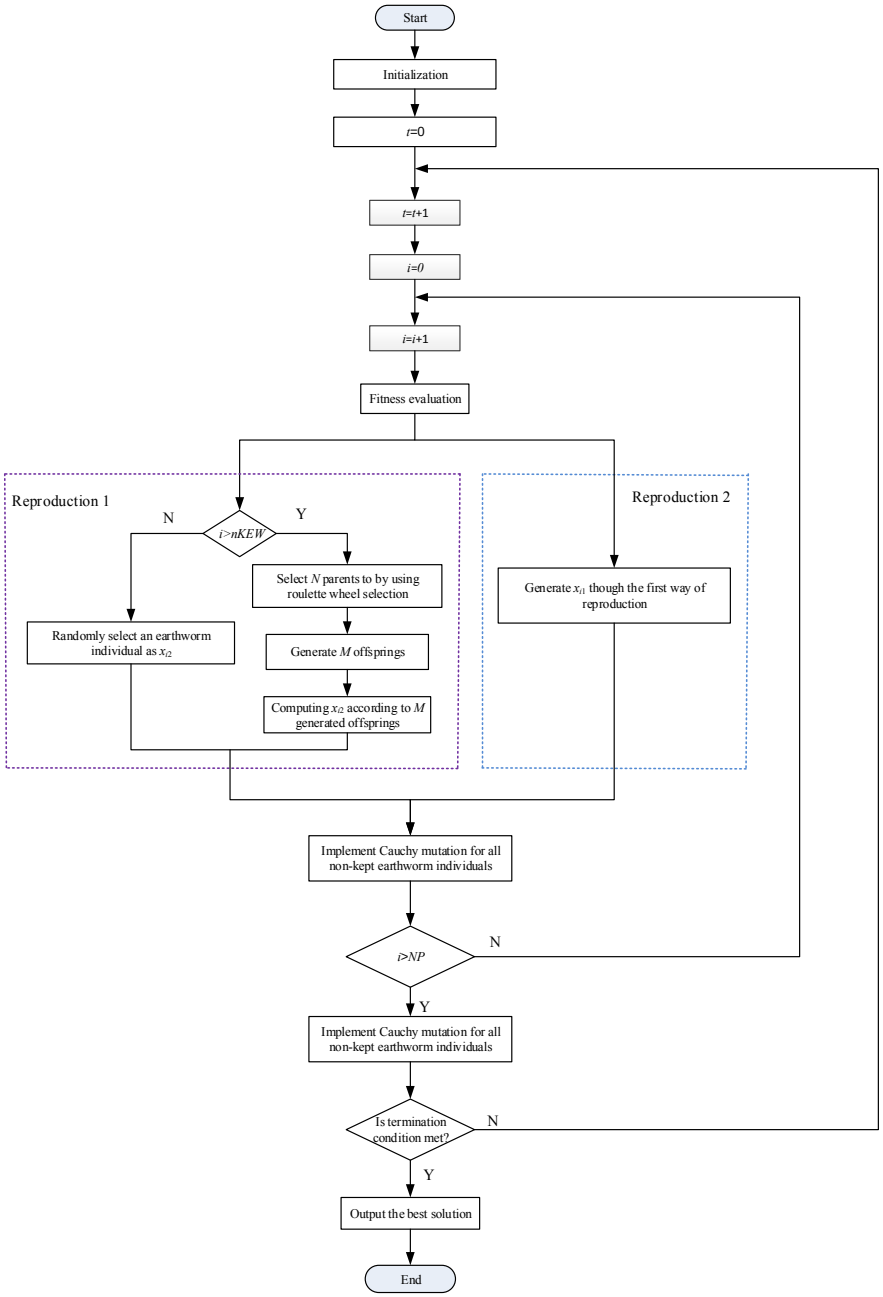
Step 5: Output the best solution.

End.

3.4 Schematic presentation of EWA algorithm

By idealising the two kinds of reproduction of the earthworm individuals, variants of EWA methods can be formed, and their schematic description can be given as shown in Algorithm 3. A brief presentation of the EWA algorithm is shown in Figure 2.

Figure 2 Schematic flowchart of EWA method (see online version for colours)



4 Simulation results

In this section, the EWA method is evaluated from various aspects by using an array of experiments conducted on benchmark functions (see Table 1) and an engineering optimisation problem. More detailed descriptions of all the benchmarks can be referred to Simon (2008), Yang et al. (2013) and Yao et al. (1999). Note that the dimensions of functions F01-F22, F23-F35, F36-F38, F39-F45 and F46-F48 are 30, two, three, four and six, respectively. That is, benchmarks F01-F22 and F23-F48 are the high-dimensional functions and low-dimensional functions, respectively.

In order to obtain fair results, all the implementations are conducted under the same conditions as shown in Wang et al. (2014f).

Table 1 Benchmark functions

No.	*Name*	*No.*	*Name*
F01	Ackley	F25	Bohachevsky #2
F02	Alpine	F26	Bohachevsky #3
F03	Brown	F27	Booth
F04	Dixon & Price	F28	Branin
F05	Fletcher-Powell	F29	Easom
F06	Griewank	F30	Foxholes
F07	Holzman 2 function	F31	Freudenstein-Roth
F08	Levy	F32	Goldstein-Price
F09	Pathological function	F33	Hump
F10	Penalty #1	F34	Matyas
F11	Penalty #2	F35	Shubert
F12	Perm	F36	Hartman1
F13	Powell	F37	Helical valley
F14	Quartic *with noise*	F38	Wolfe function
F15	Rastrigin	F39	Colville
F16	Rosenbrock	F40	Corana
F17	Schwefel 2.26	F41	Kowalik
F18	Schwefel 1.2	F42	Power Sum
F19	Schwefel 2.22	F43	Shekel #1
F20	Schwefel 2.21	F44	Shekel #2
F21	Sphere	F45	Shekel #3
F22	Step	F46	Hartman2
F23	Beale	F47	Trid06
F24	Bohachevsky #1	F48	Watson

The same parameters for variants of EWA methods are set as follows: the similarity factor α = 0.98, the initial proportional factor β_0 = 1, the constant γ = 0.9 when computing the proportional factor β. The number of generations and earthworms are set to 50 and 50, respectively. That is, $NP = MaxGen = 50$.

As we aware, metaheuristic algorithms are based on certain distribution, so 500 independent trials have been made with the aim of decreasing the influence of the randomness. In the following experiments, the optimal solution for each test problem is italic. For all the tables, the total number of functions in which the optimisation method has the best performance is provided in the last row. It is worth mentioning the scales that are used to normalise values in the tables are fully different with each other, therefore values from different tables are not comparative. The detailed normalisation process can be defined as follows.

Firstly, the real function values are presented by matrix $\mathbf{A}_{m\times n} = [\mathrm{A}_1, \mathrm{A}_2, \cdots, \mathrm{A}_i, \cdots, \mathrm{A}_m]^T$. A_i means the i_{th} row, and a_{ij} is an item in **A**, $1 \le i \le m$, $1 \le j \le n$. Here, $m = 48$ and $n = 8$ denote the number of benchmark functions and algorithms used in this paper, respectively.

Secondly, the minimum of the i^{th} row bi is calculated according to equation (38).

$$b_i = \min(\mathrm{A}_i) \tag{38}$$

Thirdly, the normalised results $C_{m\times n}$ can be obtained as shown in equation (39).

$$c_{ij} = \frac{a_{ij}}{b_i} \tag{39}$$

The values in the following tables are c_{ij} that are normalised by the above method. As an example, if matrix **A** is defined as

$$\mathbf{A} = \begin{bmatrix} 2 & 5 & 4 & 3 \\ 3 & 4 & 9 & 8 \\ 5 & 4 & 8 & 7 \end{bmatrix} \tag{40}$$

After normalised, the matrix **C** is defined as

$$\mathbf{C} = \begin{bmatrix} 1.00 & 2.50 & 2.00 & 1.50 \\ 1.00 & 1.33 & 3.00 & 2.67 \\ 1.25 & 1.00 & 2.00 & 1.75 \end{bmatrix} \tag{41}$$

4.1 The influence of the nine different crossover operators

Nine different crossover operators have been used in Reproduction 2. Therefore, nine different EWA methods can be generated accordingly. They are described as follows:

- EWA11. S1 crossover operator is used in Reproduction 2
- EWA12. M1 crossover operator is used in Reproduction 2
- EWA13. U1 crossover operator is used in Reproduction 2
- EWA21. S2 crossover operator is used in Reproduction 2
- EWA22. M2 crossover operator is used in Reproduction 2
- EWA23. U2 crossover operator is used in Reproduction 2

- EWA31. S3 crossover operator is used in Reproduction 2
- EWA32. M3 crossover operator is used in Reproduction 2
- EWA33. U3 crossover operator is used in Reproduction 2.

The results obtained by nine different EWA methods on 22 high-dimensional functions are recorded in Tables 2 to 4.

From Tables 2 to 4, it can be seen that, variants of EWA methods have the similar performance between each other though varies. The EWA method by using U2 is able to find the best solution. Looking carefully at Tables 2 to 4, the rank of performance for nine different EWA methods can be given as: EWA23 > EWA22 > EWA11 > EWA12 = EWA33 > EWA21 > EWA31 = EWA32 > EWA13 in most cases. Therefore, EWA23 is selected as the comparative algorithm in the next experiments. This shows the uniform crossover operator that generates two offsprings has a more important role on EWA method that is able to improve the EWA's performance.

4.2 Comparisons of the optimal EWA method with other methods

The performance of EWA23 was compared with seven optimisation methods [ABC (Karaboga and Basturk, 2007), ACO (Dorigo et al., 1996), BBO (Simon, 2008), DE (Storn and Price, 1997), GA (Goldberg, 1998), PSO (Kennedy and Eberhart, 1995), and SGA (Khatib and Fleming, 1998)] on 48 optimisation problems. The parameter settings for EWA23 are the same as Section 5.1. For the parameters used in the other methods, they are set as follows. For ABC, the number of colony size (employed bees and onlooker bees) NP = 50, the number of food sources $FoodNumber$ = NP / 2, maximum search times $limit$ = 100 (a food source which could not be improved through 'limit' trials is abandoned by its employed bee); for ACO, initial pheromone value τ_0 = 1E-6, pheromone update constant Q = 20, exploration constant q_0 = 1, global pheromone decay rate ρ_g = 0.9, local pheromone decay rate ρ_l = 0.5, pheromone sensitivity α = 1, and visibility sensitivity β = 5; For BBO, habitat modification probability=1, immigration probability bounds per gene = [0, 1], step size for numerical integration of probabilities = 1, maximum immigration and migration rates for each island = 1 and mutation probability = 0.005; for DE, a weighting factor F = 0.5 and a crossover constant CR = 0.5; For GA, we used roulette wheel selection, single point crossover with a crossover probability of 1, and a mutation probability of 0.01. For PSO, an inertial constant = 0.3, a cognitive constant = 1, and a social constant for swarm interaction = 1. For SGA, we used single point crossover with a crossover probability of 1, and a mutation probability of 0.01. In addition, their settings can also be referred to Simon (2008), and Wang et al. (2014e). For simplicity, EWA is short for EWA23 in the following sections. The results obtained by the eight algorithms on an array of benchmarks are recorded in Tables 5 to 7.

Table 2 Mean fitness function values obtained by EWA method with nine crossover operators

	EWA11	*EWA12*	*EWA13*	*EWA21*	*EWA22*	*EWA23*	*EWA31*	*EWA32*	*EWA33*
F01	1.16	*1.00*	1.06	1.16	1.01	1.06	1.17	1.03	1.04
F02	2.89	2.45	2.56	1.76	1.36	*1.00*	3.00	2.03	1.50
F03	1.40	1.79	2.02	*1.00*	1.61	1.57	1.08	1.47	1.49
F04	1.49	1.46	2.06	1.05	1.07	*1.00*	1.75	1.19	1.02
F05	1.07	1.19	1.15	1.17	1.17	1.16	1.26	1.24	*1.00*
F06	1.22	1.19	1.24	1.12	1.08	*1.00*	1.35	1.18	1.12
F07	13.97	2.93	3.98	2.85	*1.00*	1.07	7.62	2.08	1.25
F08	1.04	1.18	1.15	1.12	1.05	*1.00*	1.04	1.12	1.08
F09	1.09	1.05	1.09	1.06	*1.00*	1.08	1.05	1.04	1.05
F10	5.10	5.07	5.26	2.34	1.53	1.45	1.47	1.57	*1.00*
F11	2.66	2.16	4.37	2.02	2.06	*1.00*	5.17	3.72	2.49
F12	*1.00*	9.31	5.53	21.99	63.62	77.92	79.19	63.88	85.18
F13	5.08	6.22	7.22	1.63	1.42	1.30	*1.00*	1.49	1.31
F14	3.76	1.99	3.54	1.99	1.39	*1.00*	9.41	2.40	1.53
F15	1.13	1.12	1.26	1.98	1.59	*1.00*	2.18	1.42	1.34
F16	1.05	*1.00*	1.03	1.01	1.01	*1.00*	1.07	1.03	1.02
F17	1.02	1.04	*1.00*	1.04	1.07	1.05	1.05	1.08	1.03
F18	6.26	3.22	3.61	2.98	*1.00*	1.71	3.10	1.52	1.31
F19	1.15	1.07	1.08	1.10	*1.00*	*1.00*	1.37	1.07	1.14
F20	1.39	1.45	1.43	1.48	1.27	*1.00*	2.45	1.48	1.55
F21	2.44	2.08	2.45	1.73	1.34	*1.00*	3.34	1.89	1.44
F22	2.19	2.04	1.78	1.82	1.10	*1.00*	2.66	1.57	1.17
Total	1	2	1	1	4	12	1	0	2

Table 3 Best fitness function values obtained by EWA method with nine crossover operators

	EWA11	*EWA12*	*EWA13*	*EWA21*	*EWA22*	*EWA23*	*EWA31*	*EWA32*	*EWA33*
F01	1.33	*1.00*	1.12	1.24	1.18	1.21	1.32	1.19	1.27
F02	4.95	5.25	4.88	2.47	2.71	*1.00*	4.97	3.92	2.82
F03	1.41	2.97	3.64	*1.00*	2.77	2.35	1.10	2.59	2.56
F04	1.43	1.52	1.56	*1.00*	1.39	1.24	1.38	1.42	1.28
F05	1.31	1.70	1.74	1.52	2.22	2.10	1.74	2.15	*1.00*
F06	1.16	1.17	1.12	1.03	1.02	*1.00*	1.17	1.12	1.08
F07	1.90	1.65	1.75	1.63	1.27	*1.00*	3.64	*1.00*	1.12
F08	*1.00*	1.60	1.41	1.45	1.19	1.35	1.34	1.30	1.45
F09	1.20	1.17	1.05	1.13	*1.00*	1.18	1.14	1.15	1.15
F10	7.25	8.88	7.84	3.75	2.09	2.05	*1.00*	1.02	1.33
F11	2.16	3.49	7.30	3.98	2.89	*1.00*	8.85	6.47	4.69
F12	*1.00*	133.11	91.26	4.50	1.4E3	153.85	49.56	28.05	3.9E3
F13	4.58	9.26	10.47	1.41	1.73	1.96	1.53	*1.00*	1.48
F14	1.85	1.72	2.11	1.83	1.78	1.64	2.09	1.16	*1.00*
F15	1.10	*1.00*	1.36	1.69	1.82	1.01	2.23	1.30	1.34
F16	1.02	*1.00*	1.01	1.01	1.01	1.01	1.04	1.03	1.02
F17	1.10	1.10	1.10	*1.00*	1.10	1.09	1.07	1.08	1.11
F18	1.17	2.86	3.14	2.37	*1.00*	1.88	3.69	1.08	1.32
F19	1.44	1.14	1.10	1.29	*1.00*	1.25	1.52	1.36	1.33
F20	*1.00*	1.02	1.26	1.34	1.13	1.06	2.88	1.76	1.96
F21	2.42	1.98	1.71	1.49	1.31	*1.00*	1.65	1.90	1.55
F22	1.93	1.79	1.50	1.50	*1.00*	*1.00*	1.57	1.64	1.07
Total	3	3	0	3	4	6	1	2	2

Table 4 Worst fitness function values obtained by EWA method with nine crossover operators

	EWA11	*EWA12*	*EWA13*	*EWA21*	*EWA22*	*EWA23*	*EWA31*	*EWA32*	*EWA33*
F01	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*
F02	*1.00*	8.33	8.33	8.33	8.33	8.33	8.33	8.33	8.33
F03	*1.00*	1.17	9.52	9.52	9.52	9.52	9.52	9.52	9.52
F04	1.03	1.59	1.08	*1.00*	*1.00*	*1.00*	1.01	1.04	*1.00*
F05	1.31	2.25	1.74	2.05	2.79	2.95	3.60	2.18	*1.00*
F06	1.39	1.11	1.02	*1.00*	1.03	3.56	3.56	3.56	3.56
F07	2.05	18.58	1.53	9.20	*1.00*	1.70	118.79	118.79	118.79
F08	1.11	1.35	*1.00*	1.21	1.04	1.06	1.13	2.62	2.62
F09	1.20	1.23	1.10	1.16	1.08	1.10	*1.00*	1.04	1.16
F10	6.23	4.50	6.35	1.99	1.85	1.92	*1.00*	*1.00*	1.16
F11	*1.00*	16.17	16.17	16.17	16.17	16.17	16.17	16.17	16.17
F12	*1.00*	660.78	34.24	3.1E3	4.0E5	1.2E3	4.1E3	4.1E3	9.8E4
F13	6.15	9.69	7.92	1.34	1.44	1.36	1.44	1.19	*1.00*
F14	*1.00*	1.28	2.69	2.00	4.4E5	4.4E5	4.4E5	4.4E5	4.4E5
F15	*1.00*	1.14	1.40	2.46	1.33	1.13	1.58	1.35	1.75
F16	1.05	*1.00*	*1.00*	1.04	*1.00*	*1.00*	1.06	1.04	1.02
F17	1.04	*1.00*	1.10	1.07	1.03	1.08	1.16	1.11	1.11
F18	6.72	2.75	1.77	1.34	*1.00*	2.10	5.05	1.26	1.50
F19	1.14	1.09	1.11	1.01	*1.00*	1.15	1.26	*1.00*	1.27
F20	2.14	4.34	2.13	2.58	*1.00*	1.04	2.55	2.22	1.73
F21	*1.00*	1.47	56.98	56.98	56.98	56.98	56.98	56.98	56.98
F22	1.80	1.08	1.92	2.96	1.12	*1.00*	1.44	1.60	1.08
Total	8	3	3	3	7	4	3	3	4

Table 5 Mean fitness function values

	ABC	*ACO*	*BBO*	*DE*	*EWA*	*GA*	*PSO*	*SGA*
F01	4.84	5.00	3.16	4.69	*1.00*	5.26	5.06	3.43
F02	26.03	41.17	8.13	42.38	*1.00*	35.91	42.72	12.75
F03	229.41	2.0E4	17.62	33.44	*1.00*	101.11	787.62	12.15
F04	4.9E4	7.4E4	3.2E3	2.1E4	*1.00*	2.4E4	6.2E4	1.8E3
F05	3.36	9.37	*1.00*	4.07	5.18	4.79	7.77	1.23
F06	109.94	21.65	16.55	66.45	*1.00*	80.13	135.89	20.53
F07	1.5E5	2.0E5	7.1E3	6.1E4	*1.00*	7.4E4	1.3E5	6.2E3
F08	24.23	36.53	4.07	29.39	*1.00*	29.33	36.15	4.68
F09	1.40	1.32	1.16	*1.00*	1.10	1.09	1.26	1.05
F10	1.5E7	4.0E7	1.1E5	6.2E6	*1.00*	2.6E6	9.2E6	3.1E4
F11	1.6E8	4.5E8	5.0E6	6.4E7	*1.00*	4.0E7	1.1E8	2.0E6
F12	2.6E5	6.1E4	3.8E5	*1.00*	9.2E3	1.3E4	532.93	5.3E3
F13	48.04	158.94	12.07	86.99	*1.00*	46.99	67.32	5.92
F14	1.5E5	1.8E5	6.7E3	6.1E4	*1.00*	6.2E4	1.2E5	6.1E3
F15	8.85	16.00	3.37	12.53	*1.00*	13.71	12.89	5.40
F16	44.48	186.62	8.59	39.57	*1.00*	62.10	44.29	10.00
F17	2.72	1.95	*1.00*	3.29	4.01	1.67	4.55	1.24
F18	85.08	93.66	57.82	111.57	*1.00*	89.11	84.76	68.67
F19	14.12	27.20	4.67	16.14	*1.00*	20.10	32.52	7.78
F20	27.46	16.20	19.93	25.20	*1.00*	22.06	27.28	19.01
F21	348.98	638.78	50.02	211.78	*1.00*	485.60	374.23	82.13
F22	456.20	160.52	64.91	257.50	*1.00*	301.71	513.45	81.43
F23	1.01	1.03	1.04	*1.00*	1.05	1.08	1.03	1.17
F24	*1.00*	2.62	5.90	1.10	1.75	1.77	9.26	1.87
F25	*1.00*	2.21	5.41	1.08	1.46	1.37	8.54	1.75
F26	*1.00*	2.09	7.64	1.04	1.44	1.59	8.56	1.46
F27	1.04	1.05	1.19	*1.00*	1.03	1.02	1.13	1.04
F28	*1.00*	1.06	1.10	1.02	1.20	1.04	1.16	1.06
F29	2.99	2.85	3.29	3.26	*1.00*	3.15	3.11	3.11
F30	*1.00*	*1.00*	1.01	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*
F31	1.42	2.78	5.35	6.32	8.08	1.70	2.91	*1.00*
F32	1.06	1.24	1.63	1.01	*1.00*	1.44	1.19	1.39
F33	*1.00*	1.01	1.03	*1.00*	1.01	1.01	1.03	*1.00*
F34	1.01	1.01	1.01	*1.00*	*1.00*	*1.00*	1.01	1.01
F35	*1.00*	29.29	9.54	19.39	15.72	8.25	29.58	1.90
F36	1.03	3.01	3.01	*1.00*	*1.00*	3.01	1.09	3.01
F37	*1.00*	186.65	508.02	3.84	61.93	156.08	1.8E3	139.62
F38	*1.00*	1.07	*1.00*	*1.00*	*1.00*	*1.00*	1.01	*1.00*
F39	1.47	1.39	2.96	1.15	*1.00*	1.07	12.94	1.83
F40	*1.00*	1.0E6	7.4E4	47.02	1.5E3	1.8E4	5.4E5	7.3E3
F41	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	1.01	*1.00*	1.01
F42	*1.00*	2.59	3.99	3.10	2.47	16.28	4.43	22.68
F43	1.41	2.16	1.53	1.49	*1.00*	2.04	2.25	1.82
F44	1.54	2.24	2.03	1.77	*1.00*	2.25	2.33	1.94
F45	1.05	2.22	1.67	1.44	*1.00*	2.09	2.19	1.77
F46	1.15	1.70	1.22	1.11	*1.00*	1.25	1.80	1.23
F47	*1.00*	2.44	1.85	1.03	5.87	5.95	11.50	2.73
F48	19.20	11.75	14.96	21.32	*1.00*	26.34	57.23	29.70
Total	18	12	14	17	35	14	10	17

Table 6 Best fitness function values

	ABC	*ACO*	*BBO*	*DE*	*EWA*	*GA*	*PSO*	*SGA*
F01	5.63	6.55	3.68	6.43	*1.00*	6.59	6.85	4.18
F02	34.58	52.29	8.48	60.01	*1.00*	44.33	63.10	13.38
F03	107.80	8.60	13.89	42.65	*1.00*	45.17	326.03	6.82
F04	3.1E4	6.8E4	1.8E3	2.8E4	*1.00*	1.5E4	2.7E4	714.19
F05	3.71	11.73	*1.00*	5.32	5.96	4.24	9.23	*1.00*
F06	74.09	12.63	9.89	54.56	*1.00*	24.29	99.18	10.79
F07	3.7E5	5.8E5	1.5E4	2.7E5	*1.00*	1.5E5	3.7E5	1.1E4
F08	13.73	30.91	2.78	28.70	*1.00*	19.75	33.57	3.54
F09	1.65	1.51	1.35	*1.00*	1.23	1.19	1.02	1.09
F10	4.4E6	4.00	225.28	2.4E6	*1.00*	1.5E5	3.5E6	6.81
F11	1.3E8	38.10	2.9E6	8.5E7	*1.00*	9.7E6	2.6E8	1.5E4
F12	9.4E6	1.2E8	1.2E8	*1.00*	3.8E4	1.2E8	520.88	1.2E8
F13	48.94	272.97	10.78	148.40	*1.00*	16.20	49.41	4.10
F14	1.2E6	7.2E5	2.8E4	4.6E5	*1.00*	1.6E5	7.2E5	1.7E4
F15	16.21	31.20	5.25	25.42	*1.00*	22.91	27.13	7.50
F16	29.20	121.88	4.77	24.41	*1.00*	33.81	21.60	6.00
F17	3.44	2.10	*1.00*	4.37	5.01	1.40	5.67	1.23
F18	139.62	155.74	71.91	205.04	*1.00*	110.26	104.09	106.62
F19	17.77	35.47	4.73	22.86	*1.00*	27.02	30.68	6.44
F20	65.96	28.78	36.34	54.87	*1.00*	35.97	49.95	28.87
F21	568.41	1.1E3	51.93	385.77	*1.00*	511.06	598.58	84.64
F22	521.37	176.42	61.16	366.05	*1.00*	136.89	728.05	61.47
F23	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*
F24	*1.00*	*1.00*	*1.00*	*1.00*	1.09	*1.00*	1.66	*1.00*
F25	*1.00*	*1.00*	1.23	*1.00*	1.01	*1.00*	1.33	*1.00*
F26	*1.00*	*1.00*	1.04	*1.00*	*1.00*	*1.00*	1.54	*1.00*
F27	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*
F28	*1.00*	1.01	1.01	*1.00*	*1.00*	1.01	*1.00*	1.01
F29	2.96	2.57	313.35	146.45	*1.00*	80.05	36.10	5.06
F30	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*
F31	*1.00*	*1.00*	*1.00*	*1.00*	1.72	*1.00*	*1.00*	*1.00*
F32	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	1.01	*1.00*
F33	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*
F34	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*
F35	*1.00*	2.7E3	2.7E3	893.38	2.1E3	2.7E3	8.7E3	2.7E3
F36	*1.00*	3.01	3.01	*1.00*	*1.00*	3.01	*1.00*	3.01
F37	*1.00*	8.3E5	8.3E5	1.6E3	6.8E3	8.3E5	6.4E5	8.3E5
F38	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*
F39	1.13	1.06	2.10	1.17	1.02	*1.00*	6.87	1.06
F40	*1.00*	839.87	5.5E4	342.07	6.0E3	2.9E3	4.2E6	513.95
F41	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*
F42	2.1E13	2.2E14	*1.00*	2.4E14	7.3E13	*1.00*	1.1E14	3.3E14
F43	1.01	1.37	1.01	1.04	*1.00*	1.01	1.81	*1.00*
F44	*1.00*	1.69	1.01	1.06	*1.00*	1.19	1.53	*1.00*
F45	1.01	1.74	*1.00*	1.04	*1.00*	1.20	1.18	*1.00*
F46	1.03	1.32	1.15	*1.00*	*1.00*	1.15	1.41	1.15
F47	2.20	7.64	2.92	3.62	43.15	3.52	10.31	*1.00*
F48	62.30	132.05	12.01	166.48	*1.00*	29.41	437.32	15.56
Total	13	2	4	10	31	3	2	4

Table 7 Worst fitness function values

	ABC	*ACO*	*BBO*	*DE*	*EWA*	*GA*	*PSO*	*SGA*
F01	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	1.02	*1.01*	*1.00*
F02	1.14	2.00	*1.00*	1.93	*1.00*	1.55	1.82	*1.00*
F03	6.42	30.96	*1.00*	1.15	*1.00*	2.57	6.13	*1.00*
F04	7.7E3	2.2E4	1.8E3	5.9E3	*1.00*	5.2E3	1.5E4	247.15
F05	5.08	12.77	1.28	4.85	9.53	7.37	7.83	1.00
F06	79.66	21.28	9.05	36.26	*1.00*	42.05	87.99	14.16
F07	1.3E6	7.9E5	3.2E4	4.2E5	*1.00*	3.7E5	4.4E6	2.7E4
F08	24.42	40.77	3.98	27.71	*1.00*	38.10	33.05	9.82
F09	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*
F10	2.2E6	*1.00*	2.6E4	6.1E5	*1.00*	5.3E3	9.5E5	728.12
F11	7.5E6	4.6E5	2.9E5	7.3E5	*1.00*	3.0E6	7.2E6	5.2E4
F12	2.7E5	5.2E3	3.4E5	2.38	1.5E3	5.2E3	*1.00*	1.4E4
F13	32.92	319.38	32.21	108.37	*1.00*	54.45	81.37	3.37
F14	1.2E6	4.0E6	6.2E4	7.3E5	*1.00*	3.5E6	3.5E6	3.5E6
F15	8.97	19.30	3.52	13.67	*1.00*	15.29	13.29	5.40
F16	54.56	178.36	10.36	41.67	*1.00*	53.04	30.81	11.07
F17	3.22	2.07	*1.00*	3.87	4.57	1.95	5.20	1.73
F18	33.44	45.52	27.42	52.43	*1.00*	64.00	34.86	31.69
F19	2.71	5.31	*1.00*	2.87	*1.00*	3.56	6.61	1.37
F20	3.65	2.60	2.54	3.79	*1.00*	3.28	4.33	2.92
F21	4.21	8.20	*1.00*	3.55	1.72	6.63	5.04	1.72
F22	153.50	67.14	37.82	110.72	*1.00*	232.23	187.31	32.17
F23	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	1.00
F24	*1.00*	2.30	4.50	1.59	1.68	1.43	3.48	1.92
F25	*1.00*	3.25	3.20	1.09	1.49	1.24	2.52	1.13
F26	*1.00*	6.77	1.41	1.08	1.18	1.17	10.72	1.08
F27	1.03	1.02	*1.00*	1.01	1.09	1.09	1.09	1.09
F28	*1.00*	1.01	1.01	*1.00*	1.16	2.73	2.73	2.73
F29	10.89	10.89	10.88	10.89	*1.00*	10.89	11.84	11.84
F30	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*
F31	1.01	*1.00*	*1.00*	*1.00*	9.36	15.59	1.01	*1.00*
F32	*1.00*	1.97	*1.00*	1.03	1.16	*1.00*	1.18	*1.00*
F33	*1.00*	*1.00*	4.64	4.64	4.64	4.64	4.64	4.64
F34	*1.00*	*1.00*	1.01	1.59	1.59	1.59	1.59	1.59
F35	5.22	*1.00*	93.54	2.94	66.37	10.39	284.84	47.35
F36	1.05	3.01	3.01	*1.00*	*1.00*	3.01	1.07	3.01
F37	*1.00*	469.66	552.16	203.07	203.07	547.25	9.0E3	737.41
F38	*1.00*	1.10	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*
F39	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	1.04	*1.00*
F40	*1.00*	3.4E4	3.6E4	123.32	1.6E3	3.0E3	1.8E6	1.4E4
F41	*1.00*	*1.00*	16.19	16.19	16.19	16.19	16.19	16.19
F42	*1.00*	2.47	7.95	499.38	499.38	499.38	499.38	499.38
F43	1.10	1.87	1.06	1.57	*1.00*	3.20	3.20	3.20
F44	1.05	2.37	2.36	1.78	*1.00*	3.07	3.07	3.07
F45	1.91	2.71	2.27	*1.00*	1.40	2.77	3.64	3.64
F46	1.20	1.63	1.13	1.30	*1.00*	1.13	1.73	1.13
F47	1.58	6.10	*1.00*	1.66	10.62	18.80	27.76	4.35
F48	14.45	3.55	38.92	15.33	*1.00*	1.89	28.11	53.53
Total	17	11	15	10	28	6	5	11

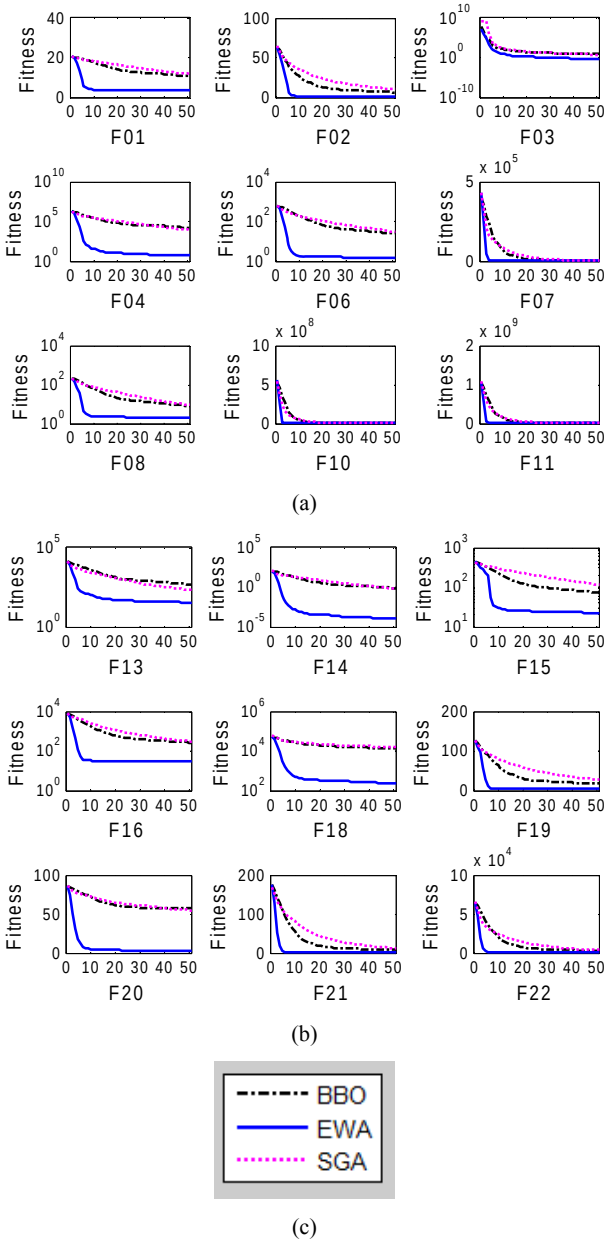
Table 8 The Std of different methods

	ABC	*ACO*	*BBO*	*DE*	*EWA*	*GA*	*PSO*	*SGA*
F01	0.89	0.88	1.04	0.55	0.54	0.90	*0.47*	1.15
F02	2.61	4.62	1.35	3.19	*0.27*	5.14	2.79	2.07
F03	72.95	4.5E4	4.07	2.60	*0.09*	26.06	271.19	1.70
F04	9.1E4	1.0E5	1.0E4	3.0E4	*2.50*	7.9E4	4.6E5	5.9E3
F05	1.6E5	3.8E5	*5.6E4*	1.5E5	2.8E5	3.2E5	2.7E5	8.5E4
F06	34.64	8.78	6.38	16.83	*0.31*	36.60	108.96	10.33
F07	2.1E4	2.9E4	2.0E3	7.3E3	*0.70*	1.8E4	7.8E4	1.6E3
F08	11.06	17.11	2.15	9.02	*0.34*	14.42	14.14	3.25
F09	*0.42*	0.52	0.56	0.70	0.55	0.68	1.03	0.78
F10	2.7E7	2.2E8	4.8E5	1.3E7	*0.98*	1.5E7	2.0E7	2.9E5
F11	5.9E7	4.7E8	3.5E6	1.9E7	*0.47*	3.1E7	4.2E7	1.7E6
F12	1.7E88	4.1E87	2.0E88	*6.2E82*	7.7E86	7.7E86	4.7E85	9.1E85
F13	546.33	1.2E3	174.83	722.97	*11.91*	673.25	705.12	124.55
F14	5.95	8.09	0.39	2.17	*2.54E-4*	5.02	4.58	0.47
F15	16.09	30.16	10.51	16.07	*7.36*	31.89	16.16	19.71
F16	339.75	821.83	68.90	227.41	*1.10*	530.26	419.38	86.31
F17	*326.10*	554.67	373.99	361.21	*645.26*	879.96	605.47	526.37
F18	3.5E3	3.4E3	3.2E3	3.1E3	*97.12*	6.2E3	4.6E3	3.8E3
F19	5.69	8.65	3.07	6.23	*0.71*	9.75	23.94	5.62
F20	5.23	6.79	8.55	4.94	*1.49*	8.46	10.37	12.95
F21	6.33	11.50	2.36	4.39	*0.10*	17.67	18.91	4.03
F22	3.7E3	1.7E3	796.26	2.0E3	*16.51*	5.4E3	9.9E3	1.1E3
F23	*0.02*	0.05	0.10	0.04	0.06	0.23	0.06	0.35
F24	*7.56E-4*	1.88	5.32	0.14	0.36	0.65	8.72	0.88
F25	*0.02*	1.84	6.25	0.09	0.27	0.45	7.17	0.93
F26	*0.01*	1.26	10.20	0.06	0.29	0.84	7.67	0.46
F27	0.08	0.07	0.24	*0.01*	0.05	0.03	0.15	0.08
F28	*3.20E-3*	0.03	0.06	0.01	0.09	0.02	0.06	0.03
F29	0.22	0.31	*0.05*	0.09	0.29	0.17	0.17	0.24
F30	*1.91E-7*	9.40E-5	0.05	1.06E-4	3.60E-3	7.08E-5	0.02	8.72E-6
F31	1.79	4.71	7.11	18.27	4.80	3.21	4.15	*0.13*
F32	1.86	1.56	5.69	0.61	*0.32*	6.03	0.87	5.96
F33	*6.88E-6*	0.01	0.04	9.76E-4	0.01	0.01	0.03	0.01
F34	0.01	0.01	0.02	7.68E-4	*2.02E-4*	0.01	0.01	0.02
F35	*1.65*	25.25	8.16	15.76	10.78	14.11	18.83	2.95
F36	2.44 E-5	1.30E-3	*3.36E-16*	3.57E-6	0.01	*3.36E-16*	0.01	*3.36E-16*
F37	*0.01*	1.98	3.35	0.03	0.40	0.74	19.99	0.74
F38	1.97E-8	0.03	2.18E-12	2.24E-13	*1.21E-14*	0.01	0.03	0.01
F39	3.89	2.90	19.33	*1.99*	2.71	3.85	37.38	10.79
F40	*1.65*	6.5E6	1.4E5	54.42	2.4E3	4.9E4	4.9E5	1.9E4
F41	1.60E-3	1.80E-3	0.01	1.40E-3	*1.30E-3*	0.01	2.90E-3	0.01
F42	0.10	0.27	0.39	0.20	*0.05*	3.29	0.33	4.13
F43	0.96	1.79	2.97	1.72	*0.19*	2.69	1.46	3.68
F44	*0.25*	1.74	3.06	1.92	2.51	1.97	1.20	3.03
F45	*1.15*	1.42	2.95	1.74	2.30	2.27	1.84	3.13
F46	*1.80E-3*	0.17	0.06	0.05	0.05	0.09	0.15	0.06
F47	3.48	7.11	8.69	*2.83*	6.42	32.17	25.64	14.44
F48	12.64	6.18	10.43	8.19	*0.77*	26.71	26.12	32.42
Total	15	0	3	4	24	1	1	2

Table 9 Comparisons between EWA and other methods at $\alpha = 0.05$ on a two-tailed t-tests

	ABC	*ACO*	*BBO*	*DE*	*GA*	*PSO*	*SGA*
F01	*79.22*	*83.44*	*39.75*	*102.67*	*87.40*	*122.04*	*40.95*
F02	*47.36*	*43.07*	*25.62*	*64.21*	*33.63*	*73.95*	*27.88*
F03	*8.84*	1.26	*11.53*	*35.17*	*10.84*	*8.19*	*18.44*
F04	*16.30*	*21.68*	*9.30*	*21.89*	*9.23*	*4.03*	*8.99*
F05	–7.88	*12.48*	–20.28	–4.86	–1.31	*9.28*	–18.73
F06	*28.76*	*21.50*	*22.26*	*35.56*	*19.77*	*11.32*	*17.27*
F07	*18.41*	*17.66*	*9.47*	*21.94*	*10.98*	*4.24*	*10.34*
F08	*25.40*	*25.11*	*17.09*	*38.05*	*23.75*	*30.05*	*13.63*
F09	*21.38*	*14.60*	*3.37*	–5.89	–0.57	*6.49*	–2.78
F10	*12.25*	*3.85*	*4.74*	*10.47*	*3.80*	*9.70*	*2.30*
F11	*14.03*	*5.06*	*7.48*	*17.92*	*6.81*	*14.08*	*6.24*
F12	*3.42*	*3.00*	*4.39*	–2.86	0.78	–2.69	–1.19
F13	*19.04*	*30.14*	*13.97*	*26.30*	*15.11*	*20.80*	*8.69*
F14	*18.27*	*15.65*	*12.10*	*19.84*	*8.67*	*19.07*	*9.19*
F15	*60.55*	*65.94*	*25.20*	*88.99*	*52.99*	*91.39*	*28.51*
F16	*24.29*	*42.88*	*20.91*	*32.20*	*21.87*	*19.60*	*19.78*
F17	–21.92	–29.81	–49.62	–11.91	–26.32	*7.52*	–40.85
F18	*35.25*	*40.00*	*25.87*	*52.22*	*21.14*	*26.84*	*26.50*
F19	*51.28*	*67.68*	*26.04*	*54.09*	*43.79*	*29.49*	*26.79*
F20	*88.50*	*39.76*	*39.64*	*85.33*	*44.56*	*45.58*	*25.12*
F21	*48.63*	*49.10*	*18.34*	*42.51*	*24.28*	*17.47*	*17.79*
F22	*31.87*	*24.88*	*20.79*	*32.63*	*14.44*	*13.45*	*19.51*
F23	–4.80	–1.25	–0.14	–4.80	1.23	–1.64	*2.78*
F24	–16.01	*3.57*	*6.03*	–12.87	0.25	*6.67*	0.99
F25	–13.37	*3.15*	*4.90*	–10.36	–1.37	*7.69*	*2.32*
F26	–11.90	*3.91*	*4.73*	–10.47	1.33	*7.22*	0.34
F27	0.13	1.19	*4.90*	–4.97	–1.93	4.77	0.68
F28	–6.78	–4.39	–2.78	–6.20	–5.19	–0.90	–4.49
F29	*12.88*	*10.20*	*18.22*	*17.41*	*15.06*	*14.65*	*13.13*
F30	–2.56	–2.51	1.80	–2.49	–2.52	*2.42*	–2.56
F31	–10.33	–6.27	–2.53	–0.74	–8.78	–6.48	–11.72
F32	0.85	*3.87*	*2.79*	0.24	1.84	*5.14*	1.65
F33	–7.17	–0.19	*3.80*	–6.02	0.50	*6.44*	–1.85
F34	*4.21*	*4.90*	*4.82*	1.97	*2.61*	*6.64*	*5.47*
F35	–7.36	*2.70*	–2.49	1.05	–2.29	*3.48*	–6.74
F36	*8.0E4*	*2.31*	0.00	5.5E5	*334.11*	0.00	*214.75*
F37	*9.03*	–1.08	–5.20	*8.84*	*4.47*	–0.76	–4.05
F38	*2.06*	*13.56*	*2.06*	*2.06*	*2.06*	1.38	–1.15
F39	*2.47*	*2.45*	*2.53*	1.12	0.39	*8.00*	1.88
F40	–5.81	1.45	*4.70*	–5.63	*3.14*	*10.06*	*2.72*
F41	0.96	*2.41*	*3.35*	–1.06	*4.29*	*4.91*	*4.04*
F42	–3.72	0.22	*2.15*	1.32	*2.64*	*3.08*	*3.08*
F43	–5.55	*8.42*	0.98	0.93	*5.81*	*10.22*	*3.03*
F44	–6.81	*7.39*	*3.98*	*2.38*	*7.08*	*9.05*	*3.28*
F45	–6.37	*8.80*	1.48	–0.61	*6.06*	*7.78*	*2.33*
F46	–13.31	*15.09*	4.20	–2.31	*4.50*	*20.23*	*4.78*
F47	–22.68	–12.18	–12.65	–23.48	0.08	*7.23*	–6.75
F48	*6.46*	*7.75*	*5.99*	*11.09*	*4.26*	*9.67*	*3.98*
Better	27	36	36	24	31	41	31
Equal	3	7	5	9	12	5	8
Worse	18	5	7	15	5	2	9

Figure 3 Convergent curves of the most preventative functions, (a) F01-F04, F06-F08 and F10-F11 (b) F13-F16 and F18-F22 (c) Legend (see online version for colours)



From Table 5, it can be seen that, on average, EWA has the best performance on 31 out of 48 benchmarks. ABC and DE have the second and third best performance on 13 and ten functions out of 48 functions, respectively. For other methods, BBO and SGA are inferior to the above methods and are able to find the best solution on four benchmarks.

For the best solutions, Table 6 shows that EWA is well capable of finding the optimal solutions on 35 out of 48 benchmarks. ABC, DE and SGA can search for the best solutions on 18, 17 and 17 out of 48 benchmarks. Generally, ACO, BBO, DE and GA have the similar performance each other. Carefully looking at the Table 6, we found that, for 22 high-dimensional complicated functions, EWA has absolute advantage over the other seven methods. While, for 13 low-dimensional functions, all the methods can solved them well under the given conditions.

For the worst function values shown in Table 7, the first three algorithms are EWA, ABC and BBO, and they perform the best on 28, 17 and 15 out of 48 benchmarks, respectively. SGA and ACO are well capable of finding the best solutions on 11 functions, which are only inferior to the above three methods.

From Tables 5 to 7, for low-dimensional benchmarks (D = 2, 3, 4 and 6 in the current work), the performance of the eight methods has little difference. Especially, ABC has the best performance when dealing with 2D functions. While, for high-dimensional benchmarks (D = 30 here), EWA is the best method at searching for the optimal function values.

Furthermore, Table 8 reveals that EWA method is well capable of optimising benchmarks with the smallest standard deviation (Std) on 24 out of 48 benchmarks among eight comparative methods. This indicates, under the same conditions, EWA is the most stable method when searching for the best solutions in most cases.

Furthermore, for the purpose of clearly showing the superiority of the EWA method, fitness curves of BBO, DSO and SGA on some most typical benchmarks are provided in this section (see Figure 3).

It can be seen in Figure 3 that, for F03, F04, F08, F14, F16, F18 and F20, EWA method is far better than BBO and SGA. While BBO and SGA have the similar performance though varies. For F01, F02, F06, F13, F15 and F19, their fitness curves have visible difference between each other. BBO has better performance than SGA on F01, F02, F06, F15 and F19, while SGA is superior to BBO for F13 case. For F07, F10, F11, F21 and F22, Figure 3 indicates that they have not clear different performance. However, from Table 5, it is observed that, EWA has the advantage of BBO and SGA.

4.3 Analyse using t-test

Based on the final search results of 500 independent trials on every function, Table 9 presents the *t* values on every function of the two-tailed test with the 5% level of significance between the EWA and other optimisation methods. In the table, the value of t with 998 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test. italicface indicates that EWA performs significantly better than the compared algorithm. For the last three rows, the 'better', 'equal', and 'worse' represent that EWA is better than, equal to, and worse than certain comparative method for this case. Here, we take the comparison between the EWA and the ABC for instance. The EWA method has far better and worse performance than ABC on 27 and 18 functions, respectively. In addition, the EWA method has the similar performance with ABC on three functions. That is to say, in general cases, the EWA outperforms ABC when solving function optimisation problems. Furthermore, for DE, the number of 'better', 'equal', and 'worse' are 24, 9 and 15, respectively. These three numbers reveal that EWA

has the better performance than DE on most functions. In other words, EWA is not worse than DE for 33 cases. Although the EWA performed equally to, or slightly poorer than the comparative methods on some functions, Table 9 indicates that it outperforms the other seven methods for most functions.

4.4 Fitness evaluations

It is well-known that, the fitness evaluations are the most time-consuming part during the optimisation process. Here, in order to fully show the advantage of the EWA method, the number of fitness evaluations is also investigated. We look at the number of fitness evaluations on 22 high-dimensional functions for each method to the fixed function values that are set to *opt* + 1. *opt* represents the minimum for each function. The maximum generation is set to 1,000, and the maximum of fitness evaluations is therefore 50,000. In other words, a method will return the fitness evaluations if it can find the function value *opt* + 1 within 50,000 generations, or return 50,000 if not (see Table 10). Table 10 reveals that, for the 13 functions, EWA can successfully find the function value *opt* + 1 by using the least fitness evaluations. SGA is only outperformed by EWA method and can successfully search for the required values by using least function evaluations on the nine functions. DE and BBO have the similar performance with each other, and they can converge to the required values with the least fitness evaluations on the seven functions and eight functions, respectively. Especially, for F07 (Holzman 2 function), F11 (Penalty #2), F14 (Quartic *with noise*) and F21 (Sphere), EWA is able to find the required function value by only using 855, 1462, 165 and 310 generations that are far less than the other seven methods. It should be mentioned that, for F05, F06, F12, F16 and F18, all the methods cannot find the required *opt* + 1 in a given situation. We have strived for finding the required values by increasing the population size and maximum generations. However, the eight methods also failed to search for the required *opt* + 1 within 500,000 fitness evaluations. On the whole, for most benchmarks, EWA has the strong ability of finding the required function values by using the least fitness evaluations.

4.5 TSC problem

Examination has a great role in motivating teaching and student learning. Automatic TSC using computer is to find a combination of questions to satisfy constraints from item bank. In essence, TSC model is a multi-constraint multi-objective optimisation problem. The description of the mathematical model for TSC problem is provided as follows (Duan et al., 2012). $K(i)$ represents that the i^{th} question is selected or not. If $K(i)$ is 0, the i_{th} question is not selected in this test-sheet, or if $K(i)$ is 1, the i^{th} question is selected in this test-sheet.

Table 10 The number of fitness evaluations for different methods

	ABC	*ACO*	*BBO*	*DE*	*EWA*	*GA*	*PSO*	*SGA*
F01	47,643	50,000	36,312	*18,055*	31,710	50,000	50,000	32,150
F02	23,327	50,000	5,245	24,745	*1,515*	33,892	50,000	5,325
F03	16,896	50,000	3,260	7,035	*1,340*	13,457	50,000	2,740
F04	49,910	50,000	48,520	27,150	*20,320*	50,000	50,000	43,947
F05	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*
F06	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*
F07	21,621	50,000	9,355	13,950	*855*	20,617	50,000	5,470
F08	18,900	49,942	4,420	12,325	5,905	16,052	50,000	*3,370*
F09	50,000	50,000	50,000	24,730	50,000	50,000	50,000	49,887
F10	21,516	46,727	8,920	17,175	5,845	21,495	50,000	5,220
F11	26,573	50,000	16,715	18,700	*1,462*	40,102	50,000	9,570
F12	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*
F13	47,748	50,000	28,802	46,645	42,865	47,022	50,000	18,160
F14	9,038	2,647	815	3,950	*165*	2,805	43,760	847
F15	50,000	50,000	*33,995*	50,000	38,052	50,000	50,000	36,750
F16	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*
F17	50,000	50,000	50,000	50,000	50,000	50,000	50,000	*43,220*
F18	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*	*50,000*
F19	25,917	50,000	*15,110*	15,180	18,532	50,000	50,000	19,422
F20	50,000	50,000	50,000	46,245	*27,685*	50,000	50,000	50,000
F21	17,193	50,000	3,782	8,570	*310*	19,595	50,000	3,467
F22	41,291	50,000	48,015	*19,715*	33,197	50,000	50,000	29,822
Total	5	5	7	8	13	5	5	9

4.5.1 Constraints

Firstly, the goal state matrix **A** is generated for all the questions:

$$\mathbf{A}_{M\times N}=\begin{pmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{pmatrix} \tag{42}$$

where M is the number of items, N is the number of attributes for each item. It does not have more than $N-1$ constraints, because ai1 (i = 1, 2, ···, M) indicates item number (Duan et al., 2012). Every objective for TSC corresponds to a constraint. Thus, the mathematical model can be given as follows (Duan et al., 2012):

1 Total score

$$\sum_{i=1}^{M} K(i)a_{i2}=S_P \tag{43}$$

where S_P is total score for the test-sheet composed; a_{i2} is score for the i^{th} question.

2 Test time

$$\sum_{i=1}^{M} K(i)a_{i3}=T_E \tag{44}$$

where T_E is the total time completing the test-sheet composed; a_{i3} is estimated time for the completion of the i^{th} question.

3 Chapter scores

$$\sum_{i=1}^{M} K(i)a_{i2}C(j)=S_C^j \tag{45}$$

where $C(j)=\begin{cases}0, & j\neq a_{i4}\\ 1, & j=a_{i4}\end{cases}$; S_C^j is the score of the j^{th} chapter; a_{i4} is the chapter number for the i^{th} question.

4 Knowledge point score

$$\sum_{i=1}^{M} K(i)a_{i2}O(j)=S_O^j \tag{46}$$

where $O(j)=\begin{cases}0, & j\neq a_{i5}\\ 1, & j=a_{i5}\end{cases}$; S_K^j is the score for the j^{th} knowledge point; a_{i5} is the knowledge point number for the i^{th} question.

5 Question type score

$$\sum_{i=1}^{M} K(i)a_{i2}T(j)=S_T^j \tag{47}$$

where $T(j)=\begin{cases}0, & j\neq a_{i6}\\ 1, & j=a_{i6}\end{cases}$; S_K^j is the score for the j^{th} question type; a_{i6} is the question type number for the i^{th} question.

6 Cognitive level score

$$\sum_{i=1}^{M} K(i)a_{i2}A(j)=S_A^j \tag{48}$$

where $A(j)=\begin{cases}0, & j\neq a_{i7}\\ 1, & j=a_{i7}\end{cases}$; S_A^j is the score for the j^{th} cognitive level; a_{i7} is the cognitive level number for the i^{th} question.

7 difficulty degree

$$\left[\sum_{i=1}^{M} K(i)a_{i2}a_{i8}\right]\Big/ S_P=DIF_P \tag{49}$$

where DIF_p is the difficulty degree of the test-sheet composed; a_{i8} is difficult degree for the i^{th} question.

8 Discrimination degree

$$\left[\sum_{i=1}^{M} K(i)a_{i2}a_{i9}\right]\Big/ S_P=DIS_P \tag{50}$$

where DIS_p is the discrimination degree of the test-sheet composed; a_{i9} is discrimination degree for the i^{th} question.

4.5.2 Objective function

In general, the error between eight constraints and evaluation requirements can be considered as objective function f. The mathematical model for the TSC problem is as follows:

$$\begin{aligned}\min f=&\sum_{j=1}^{2}\left(d_j^+ + d_j^-\right)\times\omega_j+\sum_{i=1}^{h}\left(d_{3i}^+ + d_{3i}^-\right)\times\omega_3\\ &+\sum_{i=1}^{l}\left(d_{4i}^+ + d_{4i}^-\right)\times\omega_4+\sum_{i=1}^{p}\left(d_{5i}^+ + d_{5i}^-\right)\times\omega_5\\ &+\sum_{i=1}^{q}\left(d_{6i}^+ + d_{6i}^-\right)\times\omega_6+\sum_{j=7}^{8}\left(d_j^+ + d_j^-\right)\times\omega_j\end{aligned} \tag{51}$$

$$\begin{aligned}s.t.\ &\sum_{j=1}^{M} K(j)a_{j2}-d_1^+ + d_1^- = S_P\\ &\sum_{j=1}^{M} K(j)a_{j3}-d_2^+ + d_2^- = T_E\\ &\sum_{i=1}^{h}\left(\sum_{j=1}^{M} K(j)a_{j2}C(i)-d_{3i}^+ + d_{3i}^-\right)=\sum_{i=1}^{h} S_C^i\\ &\sum_{i=1}^{l}\left(\sum_{j=1}^{M} K(j)a_{j2}O(i)-d_{4i}^+ + d_{4i}^-\right)=\sum_{i=1}^{l} S_O^i\\ &\sum_{i=1}^{p}\left(\sum_{j=1}^{M} K(j)a_{j2}T(i)-d_{5i}^+ + d_{5i}^-\right)=\sum_{i=1}^{p} S_T^i\end{aligned}$$

$$\sum_{i=1}^{q}\left(\sum_{j=1}^{M}K(j)a_{j2}A(i)-d_{6i}^{+}+d_{6i}^{-}\right)=\sum_{i=1}^{q}S_A^i$$

$$\left[\sum_{j=1}^{M}K(j)a_{j2}a_{j8}\right]\Big/S_P-d_7^{+}+d_7^{-}=DIF_P$$

$$\left[\sum_{j=1}^{M}K(j)a_{j2}a_{j9}\right]\Big/S_P-d_8^{+}+d_8^{-}=DIS_P$$

$$d_i^{+},\ d_i^{-}\geq 0,\ d_i^{+}\times d_i^{-}=0(i=1,\ 2,\ 7,\ 8)$$

$$d_{ij}^{+},\ d_{ij}^{-}\geq 0,\ d_{ij}^{+}\times d_{ij}^{-}=0(i=3,\ j=h)$$

$$d_{ij}^{+},\ d_{ij}^{-}\geq 0,\ d_{ij}^{+}\times d_{ij}^{-}=0(i=4,\ j=l)$$

$$d_{ij}^{+},\ d_{ij}^{-}\geq 0,\ d_{ij}^{+}\times d_{ij}^{-}=0(i=5,\ j=p)$$

$$d_{ij}^{+},\ d_{ij}^{-}\geq 0,\ d_{ij}^{+}\times d_{ij}^{-}=0(i=6,\ j=q)$$

$$\omega_j\geq 0(j=1,\ 2,\ ...,\ 8),\ \sum_{j=1}^{8}\omega_j=1$$

In function f, $d_i^{+}, d_{ij}^{+}, d_{ij}^{+}, d_{ij}^{+}$ and d_{ij}^{+} are the positive deviation between the test-sheet property and constraints; $d_i^{-}, d_{ij}^{-}, d_{ij}^{-}, d_{ij}^{-}$ and d_{ij}^{-} are the negative deviation between the test-sheet property and constraints. The produce of the positive deviation and negative deviation is 0, i.e., $d^{-}\times d^{+}=0$, ω_j (j = 1, 2, ···, 8) is weight for TSC problem, whose sum is 1.

In practice, the weight ω_j has an important impact on the TSC problem. In the present work, analytic hierarchy process (AHP) (Saaty, 1980) is used to address the weights. We get the goal weights for TSC model that are {0.3423, 0.1412, 0.0792, 0.0319, 0.2244, 0.0417, 0.0970, 0.0424} according to the AHP (Duan et al., 2012). In variants of EWA methods, the standard continuous encoding of EWA method cannot lend itself to address TSC problem directly. For the purpose of making the EWA method fitting for solving TSC problem, preprocessing and encoding should be implemented firstly. More details about AHP, preprocessing and encoding can be referred to Duan et al. (2012).

When dealing with the TSC problem by using EWA, the status code for each earthworm individual represents a candidate solution. Earthworm i is evaluated by the objective function $f(X_i)$ in TSC model. Here, the eight algorithms are used to find the best combination of questions. The constraints of the TSC problem are set as shown in (Duan et al., 2012).

From Table 11, we see that EWA is the most effective method on the best, mean and worst values. Moreover, EWA has the least Std among eight methods. In sum, from Table 11, EWA is well capable of finding the most satisfactory solution for TSC problem. Since TSC problem is an essentially a discrete constrained problem, therefore we can say, EWA is an encouraging method for solving discrete constrained optimisation problem, too.

5 Discussion and conclusions

By simulating the kinds of reproduction of the earthworms in nature, a new kind of bio-inspired metaheuristic algorithm, called EWA, is originally proposed for continuous and discrete constrained optimisation problems in this paper. The search direction of the earthworm individuals in EWA algorithm are mainly determined by the two kinds of reproduction. Reproduction 1 only produces one offspring by itself that is a special kind of reproduction for earthworm in nature. Reproduction 2 can generate one or more than one offsprings at one time. In EWA, the offsprings are generated through two kinds independently according to different mathematical schemes. In this way, EWA is more suitable for parallel computation and well capable of making trade-off between intensification and diversification. Subsequently, we used weighted summation of all the generated offsprings to get the final earthworm for next generation. At the same time, the incorporation of a CM is made to the EWA method in order to make the earthworm individuals escape from local optima and improve the search ability of earthworms.

Several improved crossover operators are proposed and implemented in order to generate more than one offsprings in Reproduction 2. Accordingly, nine different EWA methods with one, two and three offsprings are respectively proposed and their performance is investigated through 22 high-dimensional benchmarks. The results show that EWA23 performs the best and is further compared with seven state-of-art metaheuristic algorithms on 48 benchmark functions and an engineer case. The EWA method is able to find the better function values on most benchmark problems than seven other metaheuristic algorithms.

It is well-known that, the balance of exploration and exploitation is one of the key problems for a metaheuristic algorithm. EWA method can implement global search and local search well. This can successfully do by adjusting the similarity factor α in Reproduction 1 and the proportional factor β in Reproduction 2.

Table 11 Optimisation results for the TSC problem

	ABC	*ACO*	*BBO*	*DE*	*EWA*	*GA*	*PSO*	*SGA*
MEAN	5.4698	6.5681	4.7955	6.1197	4.5294	5.2584	6.5347	4.8051
BEST	4.0991	5.2693	3.8637	5.0246	3.4128	4.2328	5.5203	3.5501
WORST	5.4049	6.6394	5.2799	6.4168	5.1865	5.4335	6.8501	5.2816
STD	0.3191	0.3261	0.4052	0.3374	0.2800	0.4142	0.3194	0.4695

Despite various advantages of the EWA method, the following points should be clarified and focused on in the future research.

Firstly, in the current work, the performance of EWA method is experimentally tested only using benchmark problems and an engineering case. The convergence of EWA method will be analysed theoretically by dynamic systems and Markov chain. This can make variants of EWA methods implement in more stable way.

Secondly, in this paper, the number of offsprings generated in Reproduction 2 are set to 1, 2 and 3, respectively (M = 1, 2 and 3). And then, nine different crossover operators are proposed accordingly. How to propose a general frame for any number of offsprings generated in Reproduction 2 is an interesting and challenging job in the research of EWA method.

Thirdly, we only use 48 and an engineering case to test our proposed EWA method. In future, more benchmark problems and real-world applications should be used to verify the EWA method.

At last, here, the characteristics of the two kinds of reproduction are idealised to form the EWA methods. In future, more characteristics, such as bristles and shape, can be merged together with the EWA methods or idealised to generate new version of earthworm-inspired algorithm.

Acknowledgements

This work was supported by Jiangsu Province Science Foundation for Youths (No. BK20150239) and National Natural Science Foundation of China (No. 61503165 and No. 61305149).

References

Ali, M. and Pant, M. (2010) 'Improving the performance of differential evolution algorithm using Cauchy mutation', *Soft Computing*, Vol. 15, No. 5, pp.991–1007.

Bäck, T. (1996) *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, Oxford, UK.

Beyer, H. and Schwefel, H. (2002) *Natural Computing*, Kluwer Academic Publishers, Dordrecht, Netherlands.

Blakemore, R. (2001) *Tasmanian Worm Grows Second Head* [online] http://www.qvmag.tas.gov.au/zoology/invertebrata/printarchive/printtext/inv20aitems.html#20blakemore (accessed 5 October 2015).

Cai, X., Wang, L., Kang, Q. and Wu, Q. (2014) 'Bat algorithm with Gaussian walk', *International Journal of Bio-Inspired Computation*, Vol. 6, No. 3, pp.166–174.

Cui, Z., Fan, S., Zeng, J. and Shi, Z. (2013a) 'APOA with parabola model for directing orbits of chaotic systems', *International Journal of Bio-Inspired Computation*, Vol. 5, No. 1, pp.67–72.

Cui, Z., Fan, S., Zeng, J. and Shi, Z. (2013b) 'Artificial plant optimisation algorithm with three-period photosynthesis', *International Journal of Bio-Inspired Computation*, Vol. 5, No. 2, pp.133–139.

Dorigo, M., Maniezzo, V. and Colorni, A. (1996) 'Ant system: optimization by a colony of cooperating agents', *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 26, No. 1, pp.29–41.

Duan, H., Zhao, W., Wang, G. and Feng, X. (2012) 'Test-sheet composition using analytic hierarchy process and hybrid metaheuristic algorithm TS/BBO', *Mathematical Problems in Engineering*, Vol. 2012, pp.1–22.

Fong, S., Deb, S. and Yang, X-S. (2015) 'A heuristic optimization method inspired by wolf preying behavior', *Neural Computing and Applications*, Vol. 26, No. 7, pp.1725–1738.

Gandomi, A.H. and Alavi, A.H. (2011) 'Multi-stage genetic programming: a new strategy to nonlinear system modeling', *Information Sciences*, Vol. 181, No. 23, pp.5227–5239.

Gandomi, A.H. and Alavi, A.H. (2012) 'Krill herd: a new bio-inspired optimization algorithm', *Communications in Nonlinear Science and Numerical Simulation*, Vol. 17, No. 12, pp.4831–4845.

Gandomi, A.H., Talatahari, S., Tadbiri, F. and Alavi, A.H. (2013) 'Krill herd algorithm for optimum design of truss structures', *International Journal of Bio-Inspired Computation*, Vol. 5, No. 5, pp.281–288.

Gates, G. (1949) 'Regeneration in an earthworm, Eisenia foetida (Savigny) 1826. I. Anterior regeneration', *The Biological Bulletin*, Vol. 96, No. 2, pp.129–139.

Gates, G. (1953) 'On regenerative capacity of earthworms of the family Lumbricidae', *American Midland Naturalist*, Vol. 50, No. 2, pp.414–419.

Goldberg, D.E. (1998) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, New York.

Hand, D. (1992) *Genetic Programming: on the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA.

Hickman, C.P., Roberts, L.S., Larson, A., L'Anson, H. and Eisenhour, D.J. (2006) *Integrated Principles of Zoology*, McGraw-Hill, New York.

Hu, Y., Yin, M. and Li, X. (2011) 'A novel objective function for job-shop scheduling problem with fuzzy processing time and fuzzy due date using differential evolution algorithm', *The International Journal of Advanced Manufacturing Technology*, Vol. 56, No. 9, pp.1125–1138.

Karaboga, D. and Basturk, B. (2007) 'A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm', *Journal of Global Optimization*, Vol. 39, No. 3, pp.459–471.

Kennedy, J. and Eberhart, R. (1995) 'Particle swarm optimization', in *Proceeding of the IEEE International Conference on Neural Networks*, IEEE, pp.1942–1948.

Khatib, W. and Fleming, P. (1998) 'The stud GA: a mini revolution?', in Eiben, A., Back, T., Schoenauer, M. and Schwefel, H. (Eds.): *Proc. of the 5th International Conference on Parallel Problem Solving from Nature*, Springer-Verlag, New York, USA, pp.683–691.

Kirkpatrick, S., Gelatt Jr., C.D. and Vecchi, M.P. (1983) 'Optimization by simulated annealing', *Science*, Vol. 220, No. 4598, pp.671–680.

Li, X. and Yin, M. (2012) 'Self-adaptive constrained artificial bee colony for constrained numerical optimization', *Neural Computing and Applications*, Vol. 24, Nos. 3–4, pp.723–734.

Li, X. and Yin, M. (2013) 'Multiobjective binary biogeography based optimization for feature selection using gene expression data', *IEEE Transactions on NanoBioscience*, Vol. 12, No. 4, pp.343–353.

Li, X., Wang, J. and Yin, M. (2013) 'Enhancing the performance of cuckoo search algorithm using orthogonal learning method', *Neural Computing and Applications*, Vol. 24, No. 6, pp.1233–1247.

Li, X., Zhang, J. and Yin, M. (2014) 'Animal migration optimization: an optimization algorithm inspired by animal migration behavior', *Neural Computing and Applications*, Vol. 24, Nos. 7–8, pp.1867–1877.

Mirjalili, S. (2015a) 'The ant lion optimizer', *Advances in Engineering Software*, Vol. 83, pp.80–98.

Mirjalili, S. (2015b) 'Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems', *Neural Computing and Applications*, doi: 10.1007/s00521-015-1920-1.

Mirjalili, S., Mirjalili, S.M. and Hatamlou, A. (2015) 'Multi-verse optimizer: a nature-inspired algorithm for global optimization', *Neural Computing and Applications*, doi: 10.1007/s00521-015-1870-7.

Mirjalili, S., Mirjalili, S.M. and Lewis, A. (2014) 'Let a biogeography-based optimizer train your multi-layer perceptron', *Information Sciences*, Vol. 269, pp.188–209.

Mirjalili, S., Mirjalili, S.M. and Yang, X-S. (2013) 'Binary bat algorithm', *Neural Computing and Applications*, Vol. 25, Nos. 3–4, pp.663–681.

Saaty, T.L. (1980) *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*, McGraw-Hill, New York.

Saremi, S., Mirjalili, S. and Lewis, A. (2014) 'Biogeography-based optimisation with chaos', *Neural Computing and Applications*, Vol. 25, No. 5, pp.1077–1097.

Simon, D. (2008) 'Biogeography-based optimization', *IEEE Transactions on Evolutionary Computation*, Vol. 12, No. 6, pp.702–713.

Srivastava, P.R., Varshney, A., Nama, P. and Yang, X.S. (2012) 'Software test effort estimation: a model based on cuckoo search', *International Journal of Bio-Inspired Computation*, Vol. 4, No. 5, pp.278–285.

Storn, R. and Price, K. (1997) 'Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces', *Journal of Global Optimization*, Vol. 11, No. 4, pp.341–359.

Wang, G-G., Deb, S. and Cui, Z. (2015) 'Monarch butterfly optimization', *Neural Computing and Applications*, doi: 10.1007/s00521-015-1923-y.

Wang, G-G., Gandomi, A.H. and Alavi, A.H. (2014a) 'An effective krill herd algorithm with migration operator in biogeography-based optimization', *Applied Mathematical Modelling*, Vol. 38, Nos. 9–10, pp.2454–2462.

Wang, G-G., Gandomi, A.H. and Alavi, A.H. (2014b) 'Stud krill herd algorithm', *Neurocomputing*, Vol. 128, pp.363–370.

Wang, G-G., Gandomi, A.H., Yang, X-S. and Alavi, A.H. (2014c) 'A novel improved accelerated particle swarm optimization algorithm for global numerical optimization', *Engineering Computations*, Vol. 31, No. 7, pp.1198–1220.

Wang, G-G., Gandomi, A.H., Zhao, X. and Chu, H.E. (2014d) 'Hybridizing harmony search algorithm with cuckoo search for global numerical optimization', *Soft Computing*, doi: 10.1007/s00500-014-1502-7.

Wang, G-G., Guo, L., Gandomi, A.H., Hao, G-S. and Wang, H. (2014e) 'Chaotic krill herd algorithm', *Information Sciences*, Vol. 274, pp.17–34.

Wang, G., Guo, L., Wang, H., Duan, H., Liu, L. and Li, J. (2014f) 'Incorporating mutation scheme into krill herd algorithm for global numerical optimization', *Neural Computing and Applications*, Vol. 24, Nos. 3–4, pp.853–871.

Wang, H., Li, H., Liu, Y., Li, C. and Zeng, S. (2007) 'Opposition-based particle swarm algorithm with Cauchy mutation', in *IEEE Congress on Evolutionary Computation 2007 (CEC 2007)*, IEEE, pp.4750–4756.

Wang, H., Wu, Z., Rahnamayan, S., Liu, Y. and Ventresca, M. (2011) 'Enhancing particle swarm optimization using generalized opposition-based learning', *Information Sciences*, Vol. 181, No. 20, pp.4699–4714.

Wu, Q. (2011) 'Hybrid forecasting model based on support vector machine and particle swarm optimization with adaptive and Cauchy mutation', *Expert Systems with Applications*, Vol. 38, No. 8, pp.9070–9075.

Xue, F., Cai, Y., Cao, Y., Cui, Z. and Li, F. (2015) 'Optimal parameter settings for bat algorithm', *International Journal of Bio-Inspired Computation*, Vol. 7, No. 2, pp.125–128.

Yang, X.S. (2010) *Nature-inspired Metaheuristic Algorithms*, Luniver Press, Frome.

Yang, X.S. and Cui, Z. (2014) 'Bio-inspired computation: success and challenges of IJBIC', *International Journal of Bio-Inspired Computation*, Vol. 6, No. 1, pp.1–6.

Yang, X.S. and Deb, S. (2009) 'Cuckoo search via Lévy flights', in Abraham, A., Carvalho, A., Herrera, F. and Pai, V. (Eds.): *Proceeding of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009)*, IEEE Publications, Coimbatore, India, pp.210–214.

Yang, X-S., Cui, Z., Xiao, R., Gandomi, A.H. and Karamanoglu, M. (2013) *Swarm Intelligence and Bio-Inspired Computation*, Elsevier, Waltham, MA.

Yao, X., Liu, Y. and Lin, G. (1999) 'Evolutionary programming made faster', *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 2, pp.82–102.