INTERNATIONAL JOURNAL OF SYSTEMS SCIENCE, 2018
VOL. 49, NO. 1, 160–178
https://doi.org/10.1080/00207721.2017.1390702

Improved Alopex-based evolutionary algorithm by Gaussian copula estimation of distribution algorithm and its application to the Butterworth filter design

Yihang Yang, Xiang Cheng, Junrui Cheng, Da Jiang and Shaojun Li

Key Laboratory of Advanced Control and Optimization for Chemical Processes, East China University of Science and Technology, Shanghai, China

ABSTRACT
The application of evolutionary algorithms (EAs) is becoming widespread in engineering optimisation problems because of their simplicity and effectiveness. The Alopex-based evolutionary algorithm (AEA) possesses the basic characteristics of heuristic search algorithms but is lacking in adequate information about the fitness landscape of the input domain, reducing the convergence speed. To improve the performance of AEA, the Gaussian copula estimation of distribution algorithm (EDA) is embedded into the original AEA in this paper. With the help of Gaussian copula EDA, precise probability models are built utilising the best solutions, which can increase the convergence speed, and at the same time, keep the population diversity as much as possible. The simulation results on the benchmark functions and the application to the Butterworth filter design demonstrate the efficiency and effectiveness of the proposed algorithm, compared with several other EAs.

ARTICLE HISTORY
Received 20 December 2016
Accepted 2 October 2017

KEYWORDS
Evolutionary algorithm; Alopex; Gaussian copula; estimation of distribution algorithm

1. Introduction

Global optimisation problems arise in almost every field of business, engineering and science. The input domains of problems could be non-convex, disconnected and high-dimensional. Their objective functions sometimes oscillate shapely and create many competitive solutions with multimodal phenomenon (Ali, Khompatraporn, & Zabinsky, 2005). The characteristics of these problems make it difficult for conventional deterministic algorithms to optimise them. In recent two decades, evolutionary algorithms (EAs) have received considerable attention for their potential as a novel optimisation technique to address these problems (Wang & Dang, 2007). More and more EAs have been used to solve those engineering optimisation problems. For example, Rao, Savsani, and Balic proposed a teaching–learning-based optimisation algorithm to solve real parameter optimisation problems (2012). Yang, Karamanoglu, and He used a flower pollination algorithm to structural design problems (2014).

EAs are generic population-based meta-heuristic optimisation algorithms, integrating a set of evolutionary computation operators inspired by biological evolution. Genetic algorithm (GA), the first EA proposed by Holland (1975), collaborates selection, crossover and mutation operators to solve optimisation problems. Inspired by heuristic search of the original GA, lots of modified GAs and similar EAs, i.e. differential evolution (DE) (Storn & Price, 1997), particle swarm optimisation (PSO) algorithm (Kennedy & Eberhart, 1995) and artificial bee colony (ABC) algorithm (Kaeaboga, 2005), have been proposed to solve theoretical and practical problems in many fields. DE, proposed by Storn and Price (1997), is also a population-based algorithm for function optimisation. The strategy of DE is to generate a new individual by calculating vector differences between two randomly selected members in the population. The main difference between GAs and DEs is that mutation is realised by small perturbations to the genes in GAs and achieved by arithmetic combinations of individuals in DEs. PSO searches solutions, named as particles, according to its personal best solution and the current global best solution, marked as '*pbest*' and '*gbest*', respectively. ABC finds solutions, namely, food sources in ABC, by the behaviours of three kinds of artificial bees (leaders, followers and scouters). The leaders store and share food sources, the scouters search new food sources randomly and the followers find the right place of the food resource according to the shared information. ABC and PSO have been certified to be a promising method of optimisation due to its simplicity, wide applicability and outstanding performance (Kang, Li, & Li, 2013). All these computing techniques are robust to solve nonlinear optimisation problems. These algorithms do not require explicit

CONTACT Shaojun Li lishaojun@ecust.edu.cn

structure knowledge of the problems, but possess the ability to obtain multiple near-optimal solutions.

In order to escape from local optima, most of EAs use mutation operators to maintain population diversity. Besides, simulated annealing (SA) algorithm (Kirkpatrick, Gelatt, & Vecchi, 1983) can extricate solution from local optima. SA is a generic probabilistic algorithm for global optimisation problems. Here, we take the minimisation problem as the example. In SA, if a new solution is better than the old solution, the latter will be replaced, otherwise, the old one will be replaced according to a probability, which could be calculated as follows:

$$P = \exp\left(\frac{-(f(s_j) - f(s_i))}{T}\right) \tag{1}$$

where s_i is the old solution and s_j is the new one, $f(.)$ is the corresponding objective function and T is the annealing temperature. Due to the introduction of probability, the SA has the possibility to accept the worse solutions, which can help the algorithm to jump out of the local optimum.

Alopex-based evolutionary algorithm (AEA), proposed by Li and Li in 2011, is another stochastic parallel optimisation algorithm underlying the probability strategy and population intelligence. In SA, the probability is used to accept the new solution, but in AEA, the probability is used to choose the searching direction. AEA uses the parameter 'temperature', a mechanism similar to the idea of SA, to control the probability of direction selection. Although AEA can optimise complex nonlinear problems, the convergence speed is slow to some problems because of the existence of annealing mechanism. Moreover, AEA has no mechanism to identify the complex relationships between input variables. To improve the performance of AEA, Gaussian copula EDA is added into the AEA (GAEA) to extract the global statistical information and describe the coupling relations among variables. The proposed algorithm, GAEA, is evaluated on 22 benchmark functions, CEC2013 testing functions and a filter design problem. The promising results show the effectiveness of the improved algorithm for optimisation problems.

The rest of this paper is organised as follows. The fundamentals of AEA and Gaussian copula EDA are introduced in Section 2. Section 3 gives the detailed steps of the proposed algorithm, GAEA. Then, Section 4 presents the numerical simulations and analysis results by testing GAEA on 22 benchmark functions and CEC2013 testing functions. Section 5 applies the proposed GAEA to the filter design problem and the conclusions are given in Section 6.

2. Theoretical backgrounds

2.1. Alopex-based evolutionary algorithm (AEA)

The algorithm of pattern extraction (Alopex) was first proposed in 1974 (Harth & Tzanakou, 1974), which was used to solve combinatorial optimisation and pattern match problem. The main characteristic of Alopex is stochastic and heuristic search capability by using a probability strategy, so as to achieve the optimisation of target system. Based on stochastic idea of Alopex and swarm intelligence-based evolutionary mechanism, a parallel searching algorithm, named as Alopex-based evolutionary algorithm (AEA), was proposed by Li and Li (2011). In the evolutionary process of AEA, the population and its sequential transforming population (named as reference population) are utilised to generate new individuals in the next generation. For each individual in the original population and the corresponding individual in the reference population, a trail individual is created by adding (or subtracting) the weighted difference between the two individuals to (or from) the first individual, according to the probability determined by the two individuals' objective values and their locations. The trail individual is compared with the original one by one-to-one competition strategy. If the objective function of the trail individual is better, the original one will be replaced by the trail individual. The population after the replace operation will be utilised for further iteration, with a new sequential transformation of itself.

Taking an N-dimensional variable minimisation optimisation problem as an example, suppose that two populations, G_1^t and G_2^t (G_1^t contains K individuals with N variables, and G_2^t, named reference population, is the sequential transformation of G_1^t), are generated in each iteration of AEA. $X_i^t = (x_{i1}^t, x_{i2}^t \ldots, x_{iN}^t)$ and $Y_i^t = (y_{i1}^t, y_{i2}^t \ldots, y_{iN}^t)$ are the ith individuals in G_1^t and G_2^t, respectively. The evolution of X_i^t can be described as the following equations:

$$C_{ij}^t = \left|x_{ij}^t - y_{ij}^t\right| \times [F(X_i^t) - F(Y_i^t)] \tag{2}$$

$$T_j^t = \frac{1}{K}\sum_{i=1}^{K} |C_{ij}^t| \tag{3}$$

$$P_{ij}^t = \frac{1}{1 + \exp(C_{ij}^t / T_j^t)} \tag{4}$$

$$(x_{ij}^t)' = \begin{cases} x_{ij}^t + (x_{ij}^t - y_{ij}^t) \times r_1 & \text{if } P_{ij}^t > r_2 \\ x_{ij}^t - (x_{ij}^t - y_{ij}^t) \times r_1 & \text{otherwise} \end{cases} \tag{5}$$

$$X_i^{t+1} = \begin{cases} (X_i^t)' & \text{if } F\left((X_i^t)'\right) < F\left(X_i^t\right) \\ X_i^t & \text{otherwise} \end{cases} \tag{6}$$

where $i = 1, 2, \ldots, K$ (K is the number of individuals in a population) and $j = 1, 2, \ldots, N$ (N is the number of the variable dimensions). $F(.)$ is the objective function of the optimisation problem. C_{ij}^t is the correlation coefficient between x_{ij}^t and y_{ij}^t, which are the jth dimension of X_i^t and Y_i^t. T_j^t is the annealing temperature and P_{ij}^t is the probability to determine the 'evolutionary' direction of the jth dimension. r_1 and r_2 are random numbers between 0 and 1. $(x_{ij}^t)'$ is the trial variable and X_i^{t+1} is the individual reserved for the next generation.

Equations (2)–(5) are used to generate variables in each dimension of the trail individual for x_{ij}^t, based on individuals' locations and the objective values of the population. Equation (6) applies the one-to-one competition strategy to the original individuals, composing a new population G_1^{t+1} for the next iteration.

According to Equations (4) and (5), an individual X_i^t will move towards a better solution with a bigger probability ($P_{ij}^t > 0.5$) and towards the opposite direction with a smaller probability ($1 - P_{ij}^t$) in the jth dimension. Thus, the AEA has the ability to escape from the local solutions and converge to the global minimum in the evolutionary process.

It is noteworthy that the annealing temperature, T_j^t, is used to trade-off between the population diversity and the convergence speed in the evolutionary process. In the early stage of the evolutionary process, a high value of T_j^t makes the value of P_{ij}^t close to 0.5, so as to ensure a stochastic research in the initial phase of the program. In the later stage, the decreasing of T_j^t causes the polarisation of P_{ij}^t which can accelerate the convergent speed. The acceleration of the convergence may cause local optimum problems. Thus, more technique should be added to modify the original AEA, to jump out of local optimum in the late stage of evolutionary process.

2.2. Estimation of distribution algorithms

Since proposed in 1996, estimation of distribution algorithms (EDAs) (Larranaga & Lozano, 2002) have become another hot population-based heuristics search strategy. Instead of using conventional operators such as crossover and mutation, EDAs generate new individuals by sampling from an explicit probability distribution model which is constructed from promising solutions.

Generally, the evolutionary process of EDAs can be described as follows. First, an initial population is randomly generated within the search space. The objective function values of all individuals are calculated. Then a subset of individuals is selected from the initial population to establish a probability distribution model. A new population is generated by sampling from the probability distribution model. The steps above are repeated until it reaches the stop criterion.

The probability distribution models in EDAs can identify the features of promising solutions and deduce the location of better solutions, so as to evolve the population. According to the relationships between the variables, EDAs can be classified as variable-independent EDAs, bivariate correlation EDAs and multivariate correlation EDAs. They have a wide range of applications, such as in the graph matching (Cesar, Bengoetxea, Bloch, & Larranaga, 2005) and the Bayesian networks (Inza, Larranaga, Etxeberria, & Sierra, 2000). Besides, the hybrid of EDAs with other EAs can also provide a new idea for the research of optimisation algorithm, for example, Ahn, An, and Yoo (2012) combined PSO with the EDA and proposed a framework of the estimation of particle swarm distribution algorithms. Wang, Li, and Weise (2010) combined the DE with EDA to solve the economic load dispatch problem.

2.3. The multivariate Gaussian copula

Copula theory is a hot topic in statistics (Nelsen, 2007). Sklar's Theorem states that if there is a joint cumulative distribution function $F(z_1, \ldots, z_N)$ for random variables $z_1, \ldots, z_N$ which have marginal cumulative distribution functions $F_1(z_1), \ldots, F_N(z_N)$, then F can be written as a function of its margins,

$$F(z_1, \ldots, z_N) = C\,(F_1(z_1), \ldots, F_N(z_N)) \qquad (7)$$

where $F(z_1, \ldots, z_N)$ is the joint cumulative distribution function. $F_i(z_i)$is the marginal cumulative distribution of the ith variable. $C(.)$ is called a *copula*. For continuous F_i, C is unique; for discrete F_i, C is unique on $\mathrm{Ran}(F_1) \times \cdots \times \mathrm{Ran}(F_N)$, where, $\mathrm{Ran}(F_1)$ is the range of F_i. Further, if F_i and C are differentiable, the joint density $g(z_1, \ldots, z_N)$ can be written as

$$\begin{aligned} g(z_1, \ldots, z_N) = {} & g_1(z_1) \times \cdots \times g_N(z_N) \\ & \times c[F_1(z_1), \ldots, F_N(z_N)] \end{aligned} \qquad (8)$$

where $g_i(z_i)$ is the density corresponding to $F_i(z_i)$, and $c = \partial^N C/(\partial F_1 \ldots \partial F_N)$ is called the copula density. This essential result shows that the joint density can be written as a product of the marginal densities and the copula density under appropriate conditions.

There are several kinds of copula families. One of those families is the copula that underlies the multivariate Gaussian distribution. Multivariate Gaussian copula can incorporate the relationships defined by matrix $\mathbf{R}^*$. It adopts pairwise correlations among the variables to encode dependence, and its variables can be with arbitrary marginal distributions. Moreover, any positive-definite correlation matrix can be used in the Gaussian copula. The flexibility and analytical tractability of the multivariate Gaussian copula suggest that it is a promising way to deal with dependence. The relationship matrix $\mathbf{R}^*$can be Spearman's ρ or Kendall's τ or any other correlation matrix. In the Gaussian copula, the matrix $\mathbf{R}^*$ is converted into Pearson product–moment correlations. Thus, for each element of $\mathbf{R}^*$, calculate the corresponding product–moment correlation r_{ij} for the multivariate normal. If $\mathbf{R}^*$ consists of τ_{ij}, the formula is Equation (9), and if the measure is ρ_{ij}, the formula is Equation (10).

$$r_{ij} = \sin(\pi\tau_{ij}/2) \tag{9}$$

$$r_{ij} = 2\sin(\pi\rho_{ij}/6) \tag{10}$$

In this paper, the relationship matrix $\mathbf{R}^*$is Kendall's τ, and the element τ_{ij} can be calculated as

$$\tau_{ij} = \frac{2}{L(L-1)} \sum_{1\le a<b\le L} \operatorname{sign}[(\mathbf{Q}(a,i) - \mathbf{Q}(b,i))(\mathbf{Q}(a,j) - \mathbf{Q}(b,j))] \tag{11}$$

where $i, j = 1,2, \ldots, N$; L is the individual number of population $\mathbf{Q}$. The sign[$\bullet$] represents sign function which is defined as follows:

$$\operatorname{sign}\ (x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

$$c_G\left[\Phi(v_1), \ldots \Phi(v_N)\,\middle|\mathbf{R}^*\right] = \phi^{(N)}(v_1, \ldots, v_N\,|\mathbf{R})/\,[\phi(v_1) \times \cdots \times \phi(v_N)] \tag{12}$$

where Φ and ϕ denote the univariate standard normal distribution and density, respectively. Substitution of the expressions for the normal densities and algebraic manipulation lead to

$$c_G\left[\Phi(v_1), \ldots, \Phi(v_N)\,\middle|\mathbf{R}^*\right] = \exp\left\{-\mathbf{v}^{\mathrm{T}}(\mathbf{R}^{-1} - \mathbf{I})\mathbf{v}/2\right\}/|\mathbf{R}|^{1/2} \tag{13}$$

where $\mathbf{v} = (v_1, \ldots, v_N)^T$, and $\mathbf{I}$ is the $N \times N$ identity matrix.

Let c_G as a dependence function with arbitrary marginal distributions $F_1(z_1), \ldots, F_N(z_N)$. Using the normal inverse transformation Φ^{-1}, define $V_i = \Phi^{-1}[F_i(Z_i)]$ for $i = 1, \ldots, N$, and substitute these into Equations (13) and (8) to obtain the desired joint density.

$$\begin{aligned} g(z_1, \ldots, z_N\,|\mathbf{R}^*) &= g_1(z_1) \times \cdots \times g_N(z_N) \\ &\quad \times \exp\left\{-\mathbf{v}^T(\mathbf{R}^{-1} - \mathbf{I})\mathbf{v}/2\right\}/|\mathbf{R}|^{1/2} \\ &= g_1(z_1) \times \cdots \times g_N(z_N) \\ &\quad \times \exp\{-\ (\Phi^{-1}[F_1(z_1)], \ldots, \Phi^{-1}[F_N(z_N)]) \\ &\quad \times (\mathbf{R}^{-1} - \mathbf{I})(\Phi^{-1}[F_1(z_1)], \ldots, \Phi^{-1}[F_N(z_N)])^T/\ 2\} \\ &\quad /|\mathbf{R}|^{1/2} \end{aligned} \tag{14}$$

The progress for generating new individuals from the multivariate Gaussian copula probability model is as follows:

Step 1. According to Equation (11), calculate the elements of the Kendall matrix.
Step 2. Calculate the product–moment correlation matrix $\mathbf{R}$ via Equation (9).
Step 3. Establish the multiple normal distribution model, whose mean value equals zero and covariance matrix is $\mathbf{R}$. And then randomly sample from the model to generate the normal correlation matrix $\mathbf{V}$.
Step 4. According to $u_{ij} = \Phi(V_{ij})$, matrix $\mathbf{u}$ can be obtained.
Step 5. Use the inverse function $\mathbf{Z}_i = F_i^{-1}(\mathbf{u}_{:,i})$ to generate new individuals.

3. The improve AEA with Gaussian copula EDA (GAEA)

In this paper, we utilise the Gaussian copula EDA to accelerate convergence speed. The key to the problem is to establish an appropriate probability distribution model. However, earlier EDAs suppose that the variables are independent or take the relationship of variables as linear or some other simple structures, e.g. Mutual Information Maximization of Input Clustering (MIMIC) (Sun, Zhang, & Tsang, 2005) and Combining Optimizers with Mutual Information Trees (COMIT) (Baluja & Davies, 1997). However, for those variables that have a distinctly nonlinear relationship with each other, joint normal distribution cannot describe the relationship accurately. The copula can describe this kind of relationship. In this paper, Gaussian copula is selected from the copula families, with advantages of flexibility and analytical tractability (Clemen & Reilly, 1999). The marginal distribution is supposed to follow a normal distribution,

$$F_i(z_i) = \int_{-\infty}^{z_i} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(z_i-\mu_i)^2}{2\sigma_i^2}}\, dz_i \tag{15}$$

where μ_i and σ_i^2 represent the expectation and variance of variable z_i, $i = 1,2,\ldots,N$, and can be calculated with Equations (16)–(18):

$$w_j = \frac{f(j)}{\sum_{j=1}^{L} f(j)} \tag{16}$$

$$\mu_i = \frac{1}{L}\sum_{j=1}^{L}\left(w_j \times \mathbf{Q}(j,i)\right) \tag{17}$$

$$\sigma_i^2 = \frac{1}{L-1}\sum_{j=1}^{L}\left(\mathbf{Q}(j,i) - \mu_i\right)^2 \tag{18}$$

where w_j is the weighting based on fitness value, $f(j)$ is fitness value. **Q** is a population whose individuals are the L best individuals in G_1^t. And $L = SK$, where the parameter S is a selection ratio between 0 and 1.

Gaussian copula EDA describes the evolutionary tendency at the macro level through building probability distribution model, while AEA possesses the evolutionary tendency at the micro level by the pattern extraction algorithm. At the initial stage of evolutionary process, the GAEA can increase the convergence speed than AEA because the probability distribution model built by Gaussian copula EDA has the information of the best individuals. And at the late stage most of the individuals gather in a small zone, the GAEA resamples individuals from the probability distribution model, which sometimes can increase the population diversity. The optimising process of GAEA can be described as follows:

Step 1. Set the number of individuals in population (K), the number of variable dimensions (N), the selection ratio (S), replacement ratio (W), function optimisation goal, maximal function evaluation times or maximal iteration times (MIT).

Step 2. Generate a population G_1^t within the searching domain randomly and set the iteration time $t = 1$.

Step 3. Calculate the corresponding function values of individuals in the population G_1^t and rank the individuals according to the function values.

Step 4. Select $S \times K$ best individuals in G_1^t to build a subset Q^t, calculate the expectations and variances of the variables in Q^t to build each variable's marginal normal distribution $F_i(Z_i)$ according to Equations (15)–(18).

Step 5. Calculate the copula density and build the multivariate Gaussian copula probability model.

Step 6. Sample a new population including K individuals, named as G_3^t, according to the multivariate Gaussian copula probability model.

Step 7. Calculate the objective function values of individuals inG_3^t.

Step 8. Select the best $W \times K$ individuals from G_3^t to replace the worst $W \times K$ individuals in G_1^t.

Step 9. Generate the corresponding population G_2^t. Suppose the individuals in population G_1^t have a sequence number from 1 to K, randomly choose an integer k within [2, K], and then rearrange the individual order in population G_1^t according to the number sequence $[k, \ldots, K, 1, \ldots, k-1]$ to form the population G_2^t.

Step 10. Suppose that x_{ij}^t and y_{ij}^t are individual variables in G_1^t and G_2^t, respectively. Calculate the vector $x_{ij}^t - y_{ij}^t$, the difference Δf_i^t and the cross-correlation C_{ij}^t.

Step 11. Calculate the annealing temperature T_i^t and P_{ij}^t according to Equations (3) and (4). Then generate a trial variable $(x_{ij}^t)'$ according to Equation (5), and all of the trial variables make an intermediate population $(G_1^t)'$.

Step 12. By compared the objective values one-to-one between the corresponding individuals in population $(G_1^t)'$and population G_1^t, the next generation population, G_1^{t+1}, is generated according to Equation (6).

Step 13. Output the result when the termination criterion is met, otherwise, $t = t + 1$ and go to Step 4.

4. Simulation results and discussion

Twenty-two benchmark functions (Deep & Das, 2008; Gao, Liu, & Huang, 2013) are used to test the performance of GAEA. The mathematical formulas and input domains are shown in Table 1. The dimension of 22 benchmark functions is 10.

4.1. Effect of the selecting ratio and replacement ratio

The determination of the selecting ratio and replacement ratio are important for the performance of the algorithm. Here, we choose five functions (f1, f4, f8, f9 and f10) to test the influence of the parameters. These functions are all multimodal functions and the most difficult ones to get the global optimisation; thus, they are appropriate to test the performance of the algorithm.

First, we keep the replacement ratio unchanged. It is set at 0.1. The selecting ratio is set at 0.2, 0.4, 0.6 and 0.8, respectively. The GAEA is tested by optimising the five functions with 100,000 function evaluations. Each function is optimised for 30 times and the mean results are reserved.

Table 2 summarises the results with different selecting ratio values. It can be found that for the functions 1, 4 and 9, the performance of the algorithm becomes better with the increase of S except for the case of $S = 0.8$. Although for the functions 8 and 10, the algorithm performs well in

Table 1. The benchmark functions.

f	Name	Function	Bounds
f_1	Ackley	$-20\exp-(0.02\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i{}^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	[−30,30]
f_2	Cosine mixture	$0.1n+\sum_{i=1}^{n}x_i{}^2-0.1\sum_{i=1}^{n}\cos(5\pi x_i)$	[−1,1]
f_3	Exponential	$1-(\exp(-0.5\sum_{i=1}^{n}x_i{}^2))$	[−1,1]
f_4	Griewank	$1+\frac{1}{4000}\sum_{i=1}^{n}x_i{}^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$	[−600,600]
f_5	Levy and Montalvo-1	$\frac{\pi}{n}(10\sin^2(\pi y_1)+\sum_{i=1}^{n-2}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2), y_i=1+\frac{1}{4}(x_i+1)$	[−10,10]
f_6	Levy and Montalvo-2	$0.1(\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+(x_n-1)^2[1+\sin^2(2\pi x_n)])$	[−5,5]
f_7	Paviani	$45.778+\sum_{i=1}^{10}[(\ln(x_i-2))^2+(\ln(10-x_i))^2]-(\prod_{i=1}^{10}x_i)^{0.2}$	[2.001,9.999]
f_8	Rastrigin	$10n+\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)]$	[−5.12,5.12]
f_9	Rosenbrock	$\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	[−30,30]
f_{10}	Schwefel	$418.9829n-\sum_{i=1}^{n}x_i\sin(\sqrt{\lvert x_i\rvert})$	[−500,500]
f_{11}	Sinsoidal	$3.5-[2.5\prod_{i=1}^{n}\sin(x_i-\pi/6)]+\prod_{i=1}^{n}\sin[5(x_i-\pi/6)]$	[0,π]
f_{12}	Zakharov's	$\sum_{i=1}^{n}x_i^2+(\sum_{i=1}^{n}i\times x_i/2)^2+(\sum_{i=1}^{n}i\times x_i/2)^4$	[−5.12,5.12]
f_{13}	Sphere	$\sum_{i=1}^{n}x_i^2$	[−5.12,5.12]
f_{14}	Axis parallel hyper ellipsoid	$\sum_{i=1}^{n}ix_i^2$	[−5.12,5.12]
f_{15}	Schewefel-3	$\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	[−10,10]
f_{16}	Neumaier3	$n(n+4)(n-1)/6+\sum_{i=1}^{n}(x_i-1)^2-\sum_{i=2}^{n}x_ix_{i-1}$	[−100,100]
f_{17}	Salonmon	$1-\cos(2\pi\lVert x\rVert)+0.1\lVert x\rVert, \lVert x\rVert=\sqrt{\sum_{i=1}^{n}x_i{}^2}, \lVert x\rVert=\sqrt{\sum_{i=1}^{n}x_i{}^2}$	[−100,100]
f_{18}	Ellipsoidal	$\sum_{i=1}^{n}(x_i-i)^2$	[−10,10]
f_{19}	Schaffer1	$0.5+((\sin\sqrt{\sum_{i=1}^{n}x_i{}^2})^2-0.5)/(1+0.001(\sum_{i=1}^{n}x_i{}^2))^2$	[−100,100]
f_{20}	Brown3	$\sum_{i=1}^{n-1}(x_i^2)^{(x_{i+1}^2+1)}+(x_{i+1}^2)^{(x_i^2+1)}$	[−1,4]
f_{21}	New function	$\sum_{i=1}^{n}(0.2x_i^2+0.1x_i^2\sin 2x_i)$	[−10,10]
f_{22}	Cigar	$x_1^2+100,000\sum_{i=2}^{n}x_i^2$	[−10,10]

the case of $S=0.2$, the case of $S=0.6$ is the best on the whole.

Then the selecting ratio is fixed at 0.6 and the replacement ratio is set at 0.1, 0.2, 0.3, 0.4 and 0.5, the five functions with 100,000 function evaluation times are tested. Each function is optimised for 30 times and the mean results are listed in Table 3. It can be noted that in the case of $W=0.1$, the algorithm exhibits the best performance for the functions 8, 9 and 10. In the case of $W=0.2$, the algorithm provides the best performance for the functions 1 and 4. The above discussion reveals that the replacement ratio between 0.1 and 0.2 is suitable. In our experiment, the replacement ratio is set at 0.1.

4.2. Comparisons with classical EAs

A series of tests are carried out in this section, comparing GAEA with ABC algorithm (Karaboga & Basturk, 2007), DE (Storn & Price, 1997), AEA (Li & Li, 2011) and covariance matrix adaptation evolution strategy (CMAES), a typical EDA for continuous problem coded (Hansen, Muller, & Koumoutsakos, 2003). The code of ABC can be accessed at

Table 2. Results with different selecting ratio.

	0.2		0.4		0.6		0.8	
f	ABFV	SD	ABFV	SD	ABFV	SD	ABFV	SD
f_1	1.00E-01	8.69E-02	1.25E-02	3.92E-02	**7.15E-03**	**3.82E-02**	3.38E-02	7.65E-02
f_4	2.83E-02	1.56E-02	1.06E-02	2.50E-02	**1.23E-03**	**3.28E-03**	1.85E-03	5.21E-03
f_8	**1.63E+00**	**1.60E+00**	4.25E+00	3.38E+00	7.20E+00	3.70E+00	9.38E+00	3.19E+00
f_9	1.03E+00	1.72E+00	1.40E-01	7.28E-01	**5.01E-06**	**8.72E-06**	3.31E-05	7.42E-05
f_{10}	**1.27E-04**	**7.23E-10**	3.29E+00	1.07E+01	2.54E+01	4.96E+01	2.75E+01	5.02E+01

The bold values denote the value is the best one in comparison with others.

Table 3. Results with different replacement ratio.

	0.1		0.2		0.3		0.4		0.5	
f	ABFV	SD	ABFV	SD	ABFV	SD	ABFV	SD	ABFV	SD
f_1	7.15E-03	3.92E-02	**2.66E-08**	**1.01E-07**	1.00E-02	5.48E-02	9.06E-02	1.60E-01	7.62E-02	1.63E-01
f_4	1.23E-03	3.28E-03	**5.75E-04**	**2.21E-03**	3.99E-03	6.71E-03	8.39E-03	2.39E-02	1.22E-02	3.03E-02
f_8	**7.20E+00**	**3.70E+00**	8.64E+00	3.76E+00	9.37E+00	4.15E+00	9.56E+00	3.81E+00	1.15E+01	3.44E+00
f_9	**5.01E-06**	**8.72E-06**	3.28E-01	2.21E-01	6.01E+00	3.73E-01	6.56E+00	9.06E-01	6.76E+00	5.59E-01
f_{10}	**2.54E+01**	**4.96E+01**	3.23E+01	4.85E+01	3.93E+01	5.29E+01	4.89E+01	5.41E+01	8.80E+01	8.99E+01

The bold values denote the value is the best one in comparison with others.

http://mf.erciyes.edu.tr/abc/software.htm. In ABC algorithm, the population of food sources, *Food_Number*, is 100; the probabilities of food sources to be chosen, *prob*, is calculated by the fitness of the current food sources; the numbers of followers and the dimension for mutation, named as *neighbor* and *param2change*, are generated randomly. The code of DE, devec3, can be found at: http://www.icsi.berkeley.edu/∼storn/code.html#matl. The parameters in devec3 are set as follows: the DE step size, *F* is equal to 0.8; the strategy of DE algorithm is DE/rand/1/exp and the crossover probability, CR is fixed at 0.5.

In the original AEA, the population size is 100. Other parameters, such as *temperature* and *probability*, are counted automatically and change in each iteration. The code of CMAES can be found at https://www.lri.fr/∼hansen/cmaes.m. The population is fixed at 100, and the selection ratio is fixed at 0.5, without any further change in the code. Based on the experimental results in Section 4.1, in the GAEA, selection ratio (S), replacement ratio (W) and population (K) are set as 0.6, 0.1 and 100, respectively.

4.2.1. Solution qualities and the statistic tests of solutions

To test the solution qualities of the five algorithms mentioned earlier, a total of 100 independent runs for each function are conducted. The stopping criterion for the test is that 2×10^5 fitness evaluation times (FETs) are reached or there is no improvement for the best solution in consecutive 1×10^4 FETs, ensuring the convergence or near-convergence occurs in most benchmark functions. The comparison results are shown in Table 4, with the best values in boldface. In order to make a fair comparison, the initial populations are generated randomly, and the population for each algorithm is set at 100.

The indices in Table 4 are calculated as follows: rate of success (RS) counts the successful run times of the 100 runs. According to the literature (Deep & Das, 2008), a run is considered to be a success if the value obtained by the algorithm is within the setting error (0.01) of the known optimal solution. The average best function value (ABFV) and the standard deviation (SD) are accounted for the average best function values and their standard deviations in the successful runs. The statistic index, the number of no worse results (NNWR) compared to GAEA with the other four algorithms are listed at the bottom of Table 4.

The statistical NNWRs of RS, ABFV and SD in Table 4 are: 19/20/16 for GAEA, 18/9/5 for ABC, 21/10/6 for DE, 18/10/5 for AEA and 15/3/3 for CMAES. 20 ABFVs and 16 SDs for GAEA are the best in the five algorithms. Especially to the functions *f*9, *f*12–*f*15 and *f*20–*f*22, both the ABFVs and the SDs of GAEA are the smallest. Clearly, GAEA's optimisation performance outperforms those of the other four algorithms in terms of ABFV and SD as a whole, which means that GAEA can find solutions with a higher precision and a stronger stability.

It is worth noting that GAEA can find a better minimum, **−2.73E-12**, which is better than the reported best solution for *f*16 (*Neumaier-3*). The best solution is 10.000000238258737, 18.000000297585622, 24.000000386761442, 28.000000230713080, 30.000000601336406, 30.000000282266196, 28.000000032060650, 23.999999978611292, 18.000000046334858 and 10.000000225123355. However, the best solution in the current reference is 0 and the corresponding variables are 10, 18, 24, 28, 30, 30, 28, 24, 18 and 10 (Ali et al., 2005).

In general, the comparison in Table 4 indicates that the solution quality of GAEA is superior to those of AEA, DE, ABC and CMAES in most functions. To prevent the randomness of solutions for different algorithms, the sign test (Dixon & Mood, 1946) and Wilcoxon signed-rank sum test (Gibbons & Chakraborti, 2011), marked as *signtest* and *ranksum*, are utilised to assess the performance of GAEA.

For a two-sample test, the null hypothesis of *signtest* states that the two data-sets come from the same distribution, while that of *ranksum* states that the data in the former sample-set comes from a distribution with median greater than that of the latter one. Then the returned probability, p value, describes the confidence level of the null hypothesis. In this test, the former dataset is from GAEA and the other is from one of the other four algorithms. The p values of *signtest* and *ranksum* are shown

Table 4. Optimisation result of the five algorithms (goal = 0.01, $FET_{max} = 2 \times 10^5$).

	GAEA			ABC			DE			AEA			CMAES		
f	RS	ABFV	SD	RS	ABFV	SD	RS	ABFV	SD	RS	ABFV	SD	RS	ABFV	SD
f_1	95	1.86E-15	1.59E-15	**100**	7.92E-15	1.43E-15	**100**	6.53E-07	4.31E-06	64	4.55E-15	6.23E-16	94	**4.01E-16**	**1.35E-30**
f_2	**100**	**0**	**0**	**100**	**0**	**0**	**100**	**0**	**0**	**100**	**0**	**0**	**100**	7.21E-16	3.64E-31
f_3	**100**	**0**	**0**	**100**	**0**	**0**	**100**	**0**	**0**	**100**	3.33E-18	1.90E-17	**100**	2.78E-16	3.05E-32
f_4	96	1.55E-03	3.32E-03	**100**	3.20E-04	1.59E-03	39	1.93E-02	1.79E-02	86	7.34E-04	2.28E-03	99	**1.17E-15**	1.25E-30
f_5	**100**	**4.71E-32**	4.95E-47	**100**	**4.71E-32**	4.95E-47	**100**	**4.71E-32**	4.95E-47	**100**	**4.71E-32**	4.95E-47	**100**	3.43E-19	1.34E-37
f_6	**100**	**1.35E-32**	2.48E-47	**100**	**1.35E-32**	2.48E-47	**100**	**1.35E-32**	2.48E-47	**100**	**1.35E-32**	2.48E-47	**100**	3.50E-19	2.72E-37
f_7	**100**	**-4.70E-04**	**0**	**100**	**-4.70E-04**	6.90E-15	**100**	**-4.70E-04**	5.44E-20	**100**	**-4.70E-04**	3.11E-15	99	1.90E-03	1.64E-06
f_8	22	**0**	**0**	15	**0**	**0**	**100**	**0**	**0**	11	**0**	**0**	0	–	–
f_9	**100**	**4.21E-30**	**9.12E-30**	**100**	6.32E-02	6.11E-02	**100**	8.29E-08	5.49E-08	**100**	6.21E-01	9.55E-01	83	2.64E-03	1.86E-07
f_{10}	**100**	**1.27E-04**	**0**	**100**	**1.27E-04**	4.16E-13	**100**	**1.27E-04**	8.1E-20	99	**1.27E-04**	2.34E-13	0	–	–
f_{11}	**100**	**0**	**0**	**100**	4.13E-16	5.96E-16	**100**	**0**	**0**	**100**	**0**	**0**	**100**	1.72E-15	1.43E-30
f_{12}	**100**	**6.63E-115**	**4.74E-114**	0	–	–	**100**	5.55E-09	3.34E-09	**100**	4.45E-44	2.08E-43	**100**	3.18E-19	1.26E-37
f_{13}	**100**	**1.80E-185**	**0**	**100**	7.61E-104	9.97E-104	**100**	3.25E-45	3.05E-45	**100**	8.39E-101	2.43E-101	**100**	4.95E-20	9.24E-40
f_{14}	**100**	**1.71E-207**	**0**	**100**	3.35E-103	4.27E-103	**100**	1.47E-44	1.33E-44	**100**	8.16E-101	2.49E-101	**100**	6.41E-20	1.59E-39
f_{15}	**100**	**1.24E-117**	**8.83E-117**	**100**	1.00E-51	7.76E-52	**100**	3.72E-24	1.76E-24	**100**	6.29E-70	1.08E-69	**100**	2.24E-20	2.93E-41
f_{16}	**100**	**-2.73E-12**	4.46E-13	94	5.09E-01	4.04E-01	**100**	-1.74E-12	2.9E-13	**100**	2.05E-08	6.08E-08	**100**	4.49E-12	**6.35E-24**
f_{17}	**0**	--	–	**0**	–	--	**0**	--	--	**0**	--	–	**0**	--	--
f_{18}	**100**	**0**	**0**	**100**	**0**	**0**	**100**	**0**	**0**	**100**	**0**	**0**	**100**	6.18E-20	1.38E-39
f_{19}	**100**	**0.0097159**	4.77E-09	1	0.0097164	–	**100**	9.72E-03	1.43E-07	**100**	**0.0097159**	9.53E-08	**100**	3.55E-2	4.31E-05
f_{20}	**100**	**1.95E-137**	**1.95E-136**	**100**	1.31E-103	1.88E-103	**100**	1.14E-45	1.19E-45	**100**	9.24E-101	2.20E-101	**100**	5.87E-20	1.41E-39
f_{21}	**100**	**5.43E-149**	**5.43E-148**	**100**	9.59E-104	1.46E-103	**100**	4.24E-45	3.20E-45	**100**	8.79E-101	3.19E-101	**100**	5.37E-20	1.14E-39
f_{22}	**100**	**4.04E-235**	**0**	**100**	2.63E-98	4.34E-98	**100**	2.66E-40	2.62E-40	**100**	5.62E-57	8.32E-57	2	1.56E-3	2.29E-06
NNWR	19	20	16	18	9	5	21	10	6	18	10	5	15	3	3

Table 5. The *p* values of *signtest* and *ranksum*.

	GAEA vs. ABC			GAEA vs. DE			GAEA vs. AEA			GAEA vs. CMAES		
f	*signtest*	*ranksum*	*mark*	*signtest*	*ranksum*	*mark*	*signtest*	*ranksum*	*sign*	*signtest*	*ranksum*	*mark*
f_1	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	1.0000	−
f_2	−	−	NA	−	−	NA	−	−	NA	0.0000	0.0000	+
f_3	−	−	NA	−	−	NA	0.0069	0.0001	+	0.0000	0.0000	+
f_4	0.0010	1.0000	−	0.0000	0.0000	+	0.0069	0.9989	−	0.0001	1.0000	−
f_5	0.7596	0.2863	NA	−	−	NA	0.2997	0.6702	NA	0.0000	0.0000	+
f_6	0.0315	0.1471	NA	−	−	NA	0.8358	0.1508	NA	0.0000	0.0000	+
f_7	−	−	NA	−	−	NA	−	−	NA	0.0000	0.0000	+
f_8	0.0010	0.9930	−	0.0000	0.0000	+	0.0535	0.7808	NA	0.0000	0.0000	+
f_9	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+
f_{10}	0.4839	0.1484	NA	0.0000	0.0000	+	0.3681	0.9045	NA	0.0000	0.0000	+
f_{11}	0.0000	0.0000	+	0.0000	0.0000	+	−	−	NA	0.0000	0.0000	+
f_{12}	0.0000	0.0000	+	0.0000	0.0000	+	0.0002	0.0000	+	0.0000	0.0000	+
f_{13}	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+
f_{14}	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+
f_{15}	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+
f_{16}	0.0000	0.0000	+	0.0000	0.0000	+	0.0002	0.0000	+	0.0000	0.0000	+
f_{17}	0.0000	0.0000	+	0.0000	0.0000	+	0.0019	1.0000	-	0.0019	0.0000	+
f_{18}	−	−	NA	−	−	NA	−	−	NA	0.0000	0.0000	+
f_{19}	0.0000	0.0000	+	0.0000	0.0000	+	0.3681	0.0759	NA	0.0000	0.0000	+
f_{20}	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+
f_{21}	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+
f_{22}	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+	0.0000	0.0000	+

in Table 5, comparing the proposed GAEA with the other three algorithms, respectively.

As we can see in Table 5, there are three cases for the paired values of *signtest* and *ranksum*: (1) The *mark* with '+' sign, where both *p* values are less than 0.05, indicates that solutions for GAEA are significantly better than those of the compared algorithm for the designated functions. (2) The *mark* with '−' sign, where *signtest* ≤ 0.05 and *ranksum* ≥ 0.95, means that the effect of GAEA is worse than that of the other algorithm for the designated function. (3) The *mark* with 'NA' sign, where *signtest* ≥ 0.05 and *ranksum* ≤ 0.95 or *signtest* = *ranksum* = '−', indicates the effects of the two algorithms differ inconspicuously for the designated functions or the statistical tests fail.

Significantly, most of *marks* are '+' in Table 5. The horizontal comparison shows that GAEA is the best algorithm for f_2, f_3, f_5–f_7, f_9–f_{16} and f_{18}–f_{22}, without any '−' sign for *marks*. The vertical counts of '+' sign (13, 16, 11 and 20) and the counts of '−' sign (2, 0, 2 and 2) indicates that the solution quality of GAEA exceed that of DE and CMAES, slightly better than that of ABC and AEA. In a word, the statistical tests declare that the differences between GAEA and other four algorithms are statistically significant and GAEA is the winner.

4.2.2. Convergence speed and convergence stability

In order to make a further verification of the convergence performance of GAEA, three tests with different predefined permissible errors, 0.1, 0.01 and 0.001, are carried out with the corresponding maximal FETs: 1×10^5, 2×10^5 and 3×10^5, respectively. The terminal criterion for the test is that the maximal FETs are reached or a solution within the error is found. The statistical results are shown in Tables 6–8, respectively. RS counts the successful runs in 100 runs. FET_{MEAN} and FET_{SD} are the mean and deviation of FETs of the successful runs. It is noteworthy that any algorithm will stop early once it finds a solution within the error in each independent run for each function.

The NNWRs of GAEA are 21/19/18 in Table 6, 20/21/19 in Table 7 and 19/22/20 in Table 8, much better than those of the other four algorithms. In Table 6, 20 functions converge to the predefined precision in the 100 runs, three failures happen in f_1. In addition, for function f_{10}, GAEA gives a bigger average and SD of solutions than ABC. In Table 7, SDs of f_4 and f_{10} in GAEA are worse than those of ABC, but the corresponding MEANs are better. Similar phenomenon can be seen of f_{10} in Table 8. Tables 6–8 imply that GAEA, ABC, AEA and CMAES are all adaptive for solving rough results quickly. Most of FET_{MEAN} and FET_{SD} of GAEA are smaller than those of the other four algorithms. The results show that the convergence speed and convergence stability of GAEA are superior to the other four algorithms.

In order to make a clear comparison, the convergence curves of the benchmark functions are plotted in Figure 1. Note that only 20 functions, except for f_8 and f_{17}, are tested for 2×10^5 FETs until convergence because f_8 and f_{17} show a small value of RS in Tables 4–8, indicating low probability for a successful run. The abscissa

Table 6. Comparison of the convergence performance (goal $= 0.1$, $\text{FET}_{\text{max}} = 1 \times 10^5$).

	GAEA			ABC			DE			AEA			CMAES		
f	RS	FET_{MEAN} $(\times 10^2)$	FET_{SD} $(\times 10^6)$	RS	FET_{MEAN} $(\times 10^2)$	FET_{SD} $(\times 10^6)$	RS	FET_{MEAN} $(\times 10^2)$	FET_{SD} $(\times 10^6)$	RS	FET_{MEAN} $(\times 10^2)$	FET_{SD} $(\times 10^6)$	RS	FET_{MEAN} $(\times 10^2)$	FET_{SD} $(\times 10^6)$
f_1	97	163	42	100	195	28	0	–	–	98	476	126	99	160	9
f_2	100	13	1	100	41	6	100	110	18	100	31	3	100	48	1
f_3	100	3	1	100	12	3	100	31	1	100	8	2	100	20	1
f_4	100	108	43	100	118	11	17	887	79	100	237	72	100	116	22
f_5	100	14	1	100	45	6	100	138	29	100	39	4	100	54	1
f_6	100	10	1	100	32	5	100	81	1	100	24	2	100	36	1
f_7	100	22	1	100	68	8	100	160	1	100	62	4	100	78	2
f_8	20	196	33	14	544	175	16	973	163	14	402	225	6	189	70
f_9	100	181	11	83	522	143	20	987	99	0	-	-	92	488	76
f_{10}	100	249	57	100	232	37	100	446	30	100	492	67	0	-	-
f_{11}	100	15	1	100	43	10	100	140	12	100	43	4	100	49	1
f_{12}	100	25	2	2	956	44	100	729	23	100	134	8	100	110	3
f_{13}	100	13	1	100	40	5	100	106	7	100	32	3	100	46	1
f_{14}	100	17	1	100	53	6	100	138	7	100	43	3	100	59	1
f_{15}	100	28	1	100	81	5	100	206	17	100	68	3	100	90	1
f_{16}	100	32	1	2	745	280	100	521	32	100	582	92	100	142	6
f_{17}	100	44	5	0	–	–	92	834	67	100	179	23	86	133	8
f_{18}	100	20	1	100	55	6	100	106	14	100	60	3	100	50	1
f_{19}	100	19	2	100	249	95	100	305	22	100	70	8	100	59	1
f_{20}	100	14	1	100	47	5	100	82	3	100	41	3	100	39	1
f_{21}	100	13	1	100	39	5	100	109	12	100	32	2	100	47	1
f_{22}	100	42	1	100	145	7	100	317	20	100	181	8	9	226	622
NNWR	21	19	18	17	1	1	17	0	1	19	0	0	17	2	13

Table 7. Comparison of the convergence performance (goal $= 0.01$, $\mathrm{FET}_{\max} = 2 \times 10^5$).

	GAEA			ABC			DE			AEA			CMAES		
f	RS	$\mathrm{FET}_{\mathrm{MEAN}}$ $(\times 10^2)$	$\mathrm{FET}_{\mathrm{SD}}$ $(\times 10^6)$	RS	$\mathrm{FET}_{\mathrm{MEAN}}$ $(\times 10^2)$	$\mathrm{FET}_{\mathrm{SD}}$ $(\times 10^6)$	RS	$\mathrm{FET}_{\mathrm{MEAN}}$ $(\times 10^2)$	$\mathrm{FET}_{\mathrm{SD}}$ $(\times 10^6)$	RS	$\mathrm{FET}_{\mathrm{MEAN}}$ $(\times 10^2)$	$\mathrm{FET}_{\mathrm{SD}}$ $(\times 10^6)$	RS	$\mathrm{FET}_{\mathrm{MEAN}}$ $(\times 10^2)$	$\mathrm{FET}_{\mathrm{SD}}$ $(\times 10^6)$
f_1	95	160	33	100	222	23	100	1419	125	100	602	173	94	168	6
f_2	100	19	1	100	60	6	100	156	79	100	46	3	100	63	1
f_3	100	9	1	100	28	5	100	73	4	100	22	2	100	35	1
f_4	96	122	50	100	193	37	39	1841	474	98	392	141	99	134	42
f_5	100	19	1	100	64	8	100	194	13	100	54	4	100	72	1
f_6	100	16	1	100	56	7	100	127	67	100	42	3	100	56	1
f_7	100	27	2	100	91	7	100	199	36	100	82	4	99	95	1
f_8	18	288	37	16	601	164	100	1095	87	11	351	180	8	210	37
f_9	100	201	11	14	1013	411	100	1171	133	0	–	–	0	–	–
f_{10}	100	253	63	100	254	37	100	489	71	100	523	71	0	–	–
f_{11}	100	20	1	100	69	14	100	186	10	100	58	5	100	65	1
f_{12}	100	33	2	0	–	–	100	882	81	100	169	10	100	140	4
f_{13}	100	19	1	100	58	5	100	150	34	100	46	3	100	61	1
f_{14}	100	22	1	100	73	5	100	180	44	100	59	3	100	77	1
f_{15}	100	39	1	100	116	7	100	282	70	100	96	4	100	122	1
f_{16}	100	40	2	0	–	–	100	648	95	100	775	94	100	170	8
f_{17}	0	–	–	0	–	–	0	–	–	0	–	–	0	–	–
f_{18}	100	27	1	100	72	6	100	147	32	100	80	4	100	64	1
f_{19}	100	36	5	23	1124	553	100	756	85	100	159	23	100	112	7
f_{20}	100	19	1	100	65	6	100	126	60	100	57	3	100	55	1
f_{21}	100	19	1	100	57	5	100	156	55	100	48	3	100	66	1
f_{22}	100	48	1	100	163	7	100	363	13	100	216	8	2	1927	76
NNWR	20	21	19	17	1	1	21	1	1	19	0	0	15	2	14

Table 8. Comparison of the convergence performance (goal = 0.001, $FET_{max} = 3 \times 10^5$).

	GAEA			ABC			DE			AEA			CMAES		
f	RS	FET_{MEAN} ($\times 10^2$)	FET_{SD} ($\times 10^6$)	RS	FET_{MEAN} ($\times 10^2$)	FET_{SD} ($\times 10^6$)	RS	FET_{MEAN} ($\times 10^2$)	FET_{SD} ($\times 10^6$)	RS	FET_{MEAN} ($\times 10^2$)	FET_{SD} ($\times 10^6$)	RS	FET_{MEAN} ($\times 10^2$)	FET_{SD} ($\times 10^6$)
f_1	99	180	49	100	260	21	92	1546	154	99	678	180	95	200	9
f_2	100	24	1	100	77	6	100	201	57	100	59	3	100	79	1
f_3	100	14	1	100	44	6	100	114	62	100	36	3	100	50	1
f_4	86	130	53	100	276	154	99	2221	309	83	615	485	100	150	74
f_5	100	24	1	100	82	9	100	233	74	100	68	5	100	90	1
f_6	100	22	1	100	74	8	100	174	23	100	61	5	100	77	1
f_7	100	32	2	100	111	9	100	238	33	100	100	5	100	108	1
f_8	23	217	28	17	423	126	100	1155	85	0	–	–	4	235	48
f_9	99	219	13	0	–	–	98	1321	137	0	–	–	0	–	–
f_{10}	100	246	50	100	276	33	100	534	26	100	585	85	0	–	–
f_{11}	100	26	1	100	98	14	100	226	12	100	74	4	100	80	1
f_{12}	100	41	2	0	–	–	100	1103	89	100	207	12	100	169	3
f_{13}	100	24	1	100	77	7	100	190	37	100	61	3	100	77	1
f_{14}	100	28	1	100	89	7	100	218	53	100	74	4	100	94	1
f_{15}	100	49	1	100	152	9	100	368	62	100	123	4	100	154	1
f_{16}	100	46	2	0	–	–	100	726	83	100	974	116	100	201	10
f_{17}	0	–	–	0	–	–	0	–	–	0	–	–	0	–	–
f_{18}	100	33	1	100	92	7	100	183	58	100	98	4	100	78	1
f_{19}	0	–	–	0	–	–	0	–	–	0	–	–	0	–	–
f_{20}	100	25	1	100	85	7	100	176	52	100	73	4	100	70	1
f_{21}	100	25	1	100	76	7	100	199	73	100	65	3	100	81	1
f_{22}	100	53	1	100	182	8	100	407	54	100	247	9	1	2973	–
NNWR	19	22	20	18	2	2	18	2	3	18	2	2	17	2	15

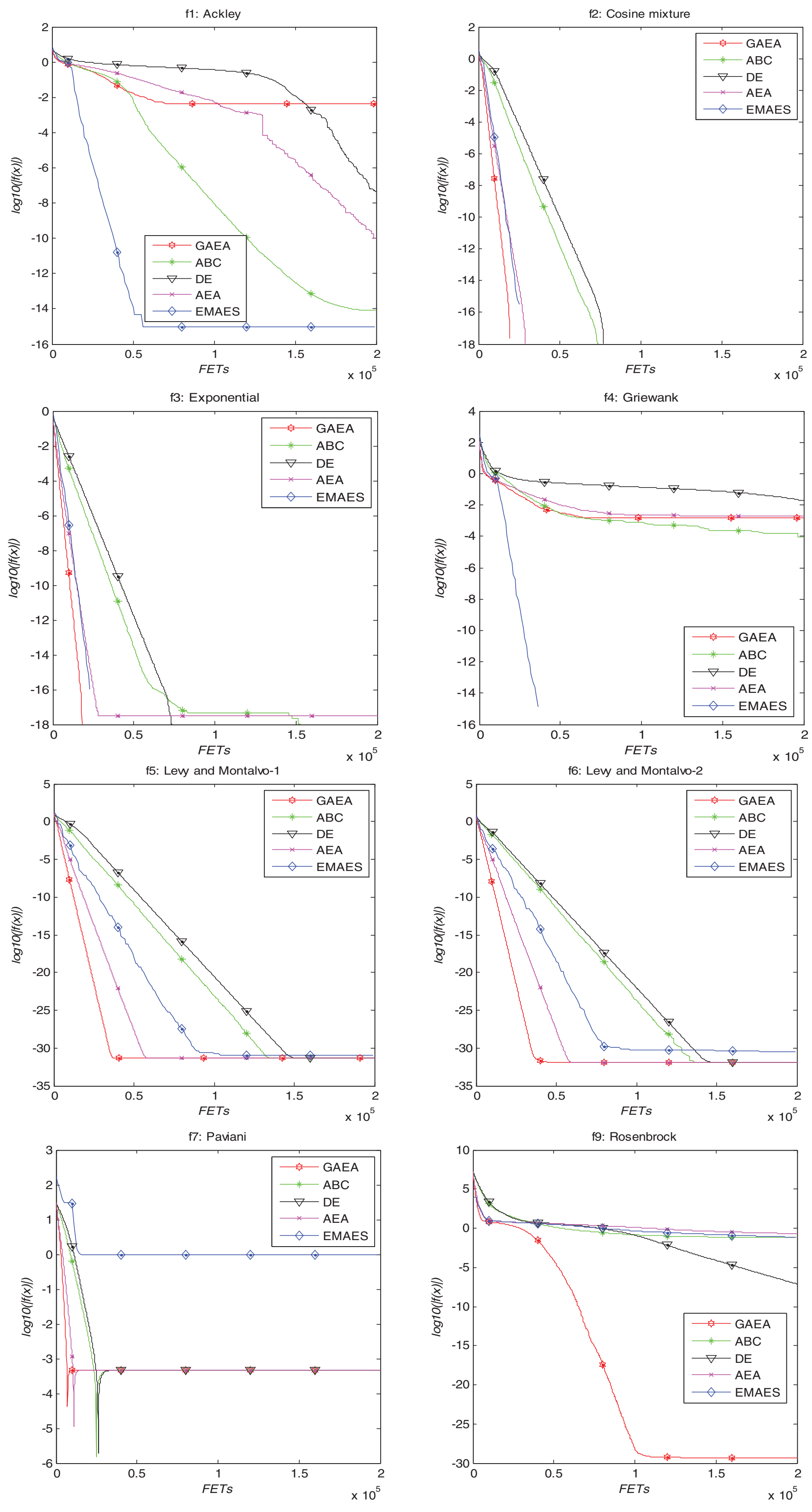

Figure 1. The convergence curves of 20 functions.

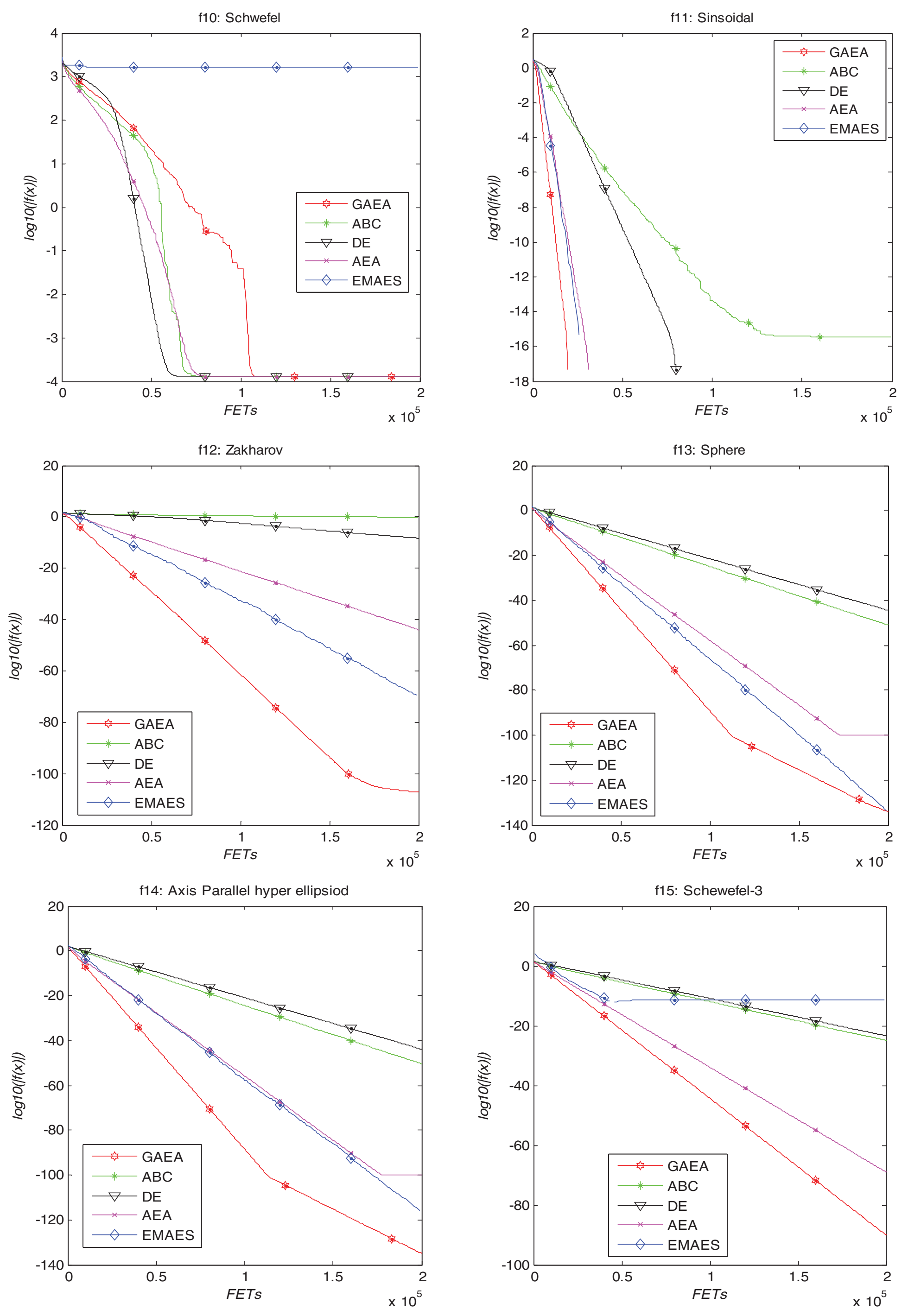

Figure 1. (*Continued*)

represents FETs; and the ordinate represents the logarithm of the absolute value of objective function.

In Figure 1, all curves drop except those of f_7 and f_{16}. Consulting the mathematical functions, the minimum of f_7 is negative (−4.70E-04), which explain why curves drop first and then rises. The curves of f_{16} show that only GAEA and DE rise, indicating that only GAEA and DE can find a negative solution.

CMAES shows the best convergence curves in f_1 and f_4 on the solution quality and the convergence speed, indicating that CMAES are suitable for simple optimisation problem without multivariable correlativity. However, CMAES shows worse effects than GAEA in other functions. For f_{10}, ABC and AEA are superior to GAEA in the convergence speed, while CMAES shows worst solution quality. The rest of the figures show that GAEA is

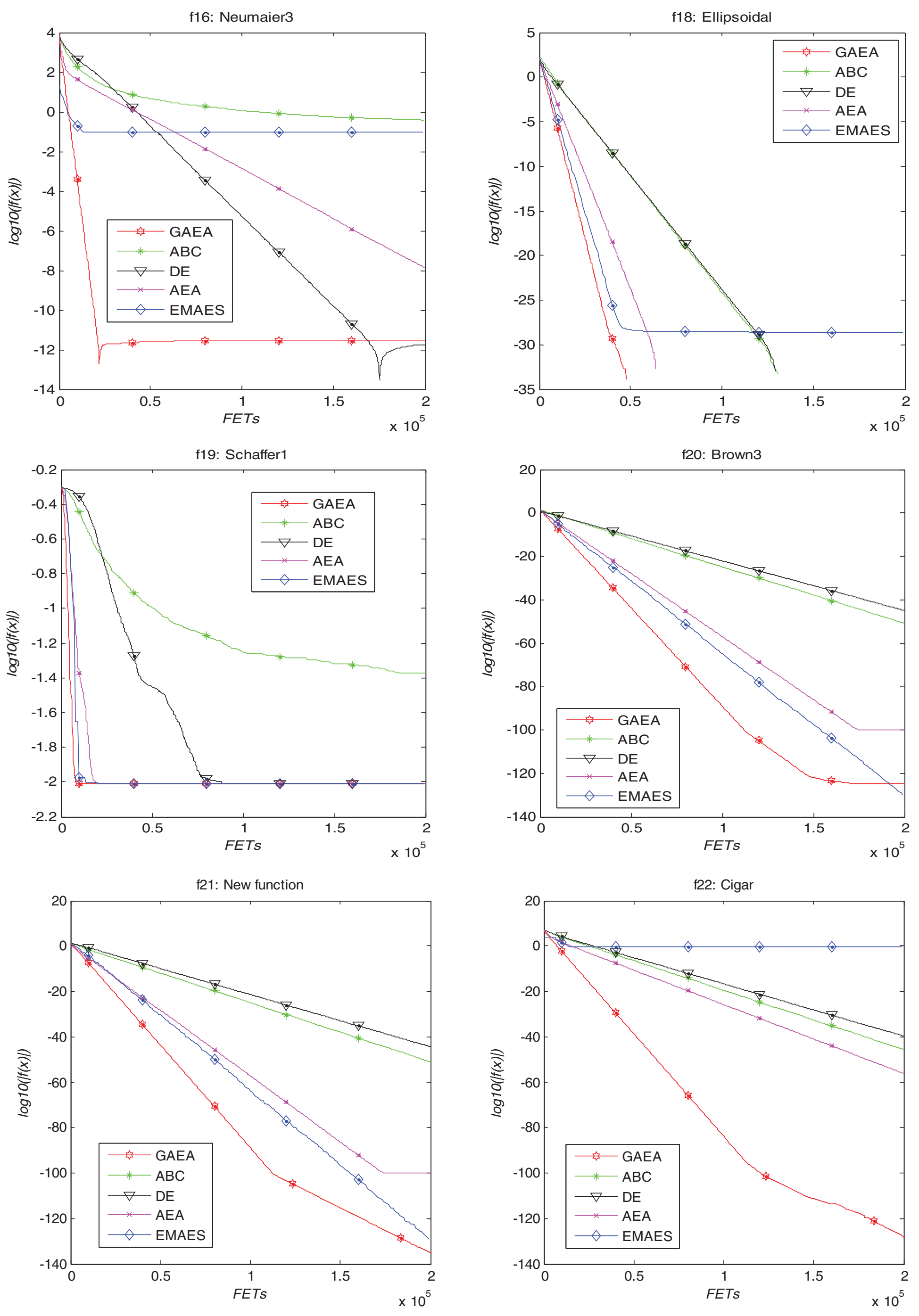

Figure 1. (*Continued*)

better than the other four algorithms in both the convergence precision and the convergence speed.

By analysing the optimising results of benchmark functions, the introduction of Gaussian copula EDA helps AEA algorithm to find the correlation of the input variables, such as in f_9, f_{16} and f_{20}, so as to converge quickly. In addition, with the increasing of the function evaluations, convergent results of GAEA decrease fast. The cause of faster convergent speed of GAEA is that the algorithm makes use of gradient-like information and relationship among variables.

4.3. Comparison with ARFO on CEC2013 benchmark functions

In this section, the GAEA is compared with an artificial root foraging optimisation (ARFO) algorithm (Ma, Zhu, Liu, Tian, & Chen, 2015), which had been tested and

Table 9. Comparison between GAEA and ARFO on CEC2013 benchmark functions.

		ARFO				GAEA			
Fun	Optimal	Best	Worst	Mean	SD	Best	Worst	Mean	SD
F1	−1400	**− 1.400E+03**	**− 1.400E+03**	**− 1.400E+03**	**0.000E+00**	**− 1.400E+03**	**− 1.400E+03**	**− 1.400E+03**	**0.000E+00**
F2	−1300	**− 1.300E+03**	**− 1.300E+03**	**− 1.300E+03**	**0.000E+00**	− 5.677E+02	2.330E+04	6.546E+03	6.642E+03
F3	−1200	**− 1.200E+03**	**− 1.200E+03**	**− 1.200E+03**	**1.411E+00**	**− 1.200E+03**	8.698E+02	− 1.124E+03	3.777E+02
F4	−1100	**− 1.100E+03**	**− 1.100E+03**	**− 1.100E+03**	**0.000E+00**	− 1.004E+03	1.022E+03	− 5.248E+02	5.159E+02
F5	−1000	**− 1.000E+03**	**− 1.000E+03**	**− 1.000E+03**	2.356E−11	**− 1.000E+03**	**− 1.000E+03**	**− 1.000E+03**	**0.000E+00**
F6	−900	**− 9.000E+02**	**− 8.960E+02**	**− 8.998E+02**	**8.914E−01**	**− 9.000E+02**	− 8.902E+02	− 8.951E+02	4.990E+00
F7	−800	− 7.345E+02	− 6.723E+02	− 7.144E+02	1.342E+01	**− 8.000E+02**	**− 7.999E+02**	**− 8.000E+02**	**1.855E-02**
F8	−700	− 6.796E+02	− 6.791E+02	− 6.794E+02	6.603E**-02**	**− 6.798E+02**	**− 6.795E+02**	**− 6.797E+02**	**5.555E-02**
F9	−600	− 5.964E+02	− 5.832E+02	− 5.905E+02	3.680E+00	**− 6.000E+02**	**− 5.981E+02**	**− 5.992E+02**	**6.434E-01**
F10	−500	**− 5.000E+02**	**− 5.000E+02**	**− 5.000E+02**	**2.036E−02**	**− 5.000E+02**	-4.996E+02	**− 5.000E+02**	6.532E-02
F11	−400	− 3.930E+02	2.884E+02	− 2.884E+02	2.022E+02	**− 4.000E+02**	**− 3.939E+02**	**− 3.986E+02**	**1.703E+00**
F12	−300	− 2.920E+02	5.079E+02	− 3.700E+01	3.308E+02	**− 3.000E+02**	**− 2.804E+02**	**− 2.934E+02**	**5.236E+00**
F13	−200	− 1.761E+02	1.204E+03	1.416E+02	3.834E+02	**− 2.000E+02**	**− 1.837E+02**	**− 1.953E+02**	**4.350E+00**
F14	−100	1.165E+03	2.330E+03	1.741E+03	3.167E+02	**4.566E+02**	**9.922E+02**	**7.728E+02**	**1.335E+02**
F15	100	7.050E+03	1.012E+04	8.604E+03	5.563E+02	**4.171E+02**	**1.469E+03**	**1.211E+03**	**2.153E+02**
F16	200	**2.000E+02**	**2.003E+02**	**2.002E+02**	**8.771E-02**	2.006E+02	2.016E+02	2.012E+02	2.284E-01
F17	300	**3.148E+02**	1.469E+03	4.363E+02	3.419E+02	3.166E+02	**3.299E+02**	**3.256E+02**	**2.532E+00**
F18	400	**4.167E+02**	1.069E+03	4.570E+02	1.441E+02	4.212E+02	**4.349E+02**	**4.289E+02**	**3.739E+00**
F19	500	**5.005E+02**	5.022E+02	**5.012E+02**	5.054E−01	5.006E+02	**5.020E+02**	5.015E+02	**3.183E-01**
F20	600	**6.000E+02**	**6.010E+02**	**6.005E+02**	1.235E+00	6.013E+02	6.029E+02	6.020E+02	**3.893E-01**
F21	700	**8.000E+02**	**1.100E+03**	**1.085E+03**	6.713E+01	1.100E+03	1.100E+03	1.100E+03	**4.625E-13**
F22	800	1.906E+03	3.689E+03	3.096E+03	4.420E+02	**9.881E+02**	**1.897E+03**	**1.371E+03**	**2.608E+02**
F23	900	**9.000E+02**	8.543E+03	**1.400E+03**	2.234E+03	9.128E+02	**1.998E+03**	1.429E+03	**2.587E+02**
F24	1000	**1.113E+03**	1.628E+03	1.214E+03	1.635E+03	1.200E+03	**1.200E+03**	**1.200E+03**	6.657E-04
F25	1100	**1.300E+03**	1.334E+03	1.305E+03	5.212E+01	**1.300E+03**	**1.300E+03**	**1.300E+03**	**1.594E-02**
F26	1200	1.304E+03	1.561E+03	1.424E+03	8.254E+01	**1.301E+03**	**1.400E+03**	**1.334E+03**	**4.216E+01**
F27	1300	**1.558E+03**	1.871E+03	**1.581E+03**	7.414E+01	1.600E+03	**1.600E+03**	1.600E+03	**5.503E-02**
F28	1400	1.800E+03	2.203E+03	1.902E+03	9.323E+02	**1.700E+03**	**1.700E+03**	**1.700E+03**	**0.000E+00**
NNWR		**17**	10	13	7	**17**	**20**	**18**	**21**

The bold values denote the value is the best one in comparison with others.

demonstrated to have a better performance than SPSO-2011 and several other state-of-the-art algorithms. In this test, the CEC2013 benchmark functions are adopted to test the proposed algorithm. The CEC2013 functions, including 5 unimodal functions (F1-T5), 15 multimodal functions (F6–F20) and 8 composition functions (F21–F28), are more complicated and challenging than the traditional ones.

Here we set the stop criteria as 10,000 D function evaluation times. The population size is set at 100. The selecting ratio and replacement ratio are set to the same with the case in Section 4.3. The results, including the best, worst, mean and SD, are listed in Table 9. It can be observed that ARFO can find the global optima for all the five unimodal functions whereas GAEA can find the optima only for two functions (F1 and F5). However, for the multimodal functions, F7–F9 and F11–F15, GAEA performs better than ARFO. For the eight composition ones, GAEA totally outperforms ARFO on four functions (F22, F25, F26 and F28). According to Table 9, the NNWR values of the best, worst, average solutions and the variance of solutions of ARFO and GAEA are 17/10/13/7 and 17/20/18/21. Especially, the worst, average and variance of solutions of GAEA are far better than those of ARFO. So, from the results above, we can find that GAEA algorithm is a competitive algorithm.

5. Applying GAEA to the optimal Butterworth filter design problem

The Butterworth filter is a type of signal processing filter designed to have the maximal flatness in its frequency response in the pass band (Butterworth, 1930). The task of the Butterworth filter design problems is to solve proper values for the circuit components, with a reputation for optimising 'impossible' mathematical problems, because of a vast of local optima.

A fourth-order Butterworth filter is composed by two second-order cascading Butterworth filter sections, as shown in Figure 2. Where R_a, R_b, C_a and C_b are the circuit components for the first section. R_c, R_d, C_c and C_d are the circuit components for the second section.

The overall transfer function is given by the product of the transfer functions of the two sections, as shown in Equation (19):

$$\begin{aligned} H(s) &= H_1(s) \times H_2(s) \\ &= \frac{\omega^2_{\text{cut-off1}}}{s^2 + s\frac{\omega_{\text{cut-off1}}}{Q_{b1}} + \omega^2_{\text{cut-off1}}} \times \frac{\omega^2_{\text{cut-off2}}}{s^2 + s\frac{\omega_{\text{cut-off2}}}{Q_{b2}} + \omega^2_{\text{cut-off2}}} \end{aligned} \tag{19}$$

where $H_1(s)$, $\omega_{\text{cut-off1}}$ and Q_{b1} are the transfer function, the cut-off frequency and the corresponding quality

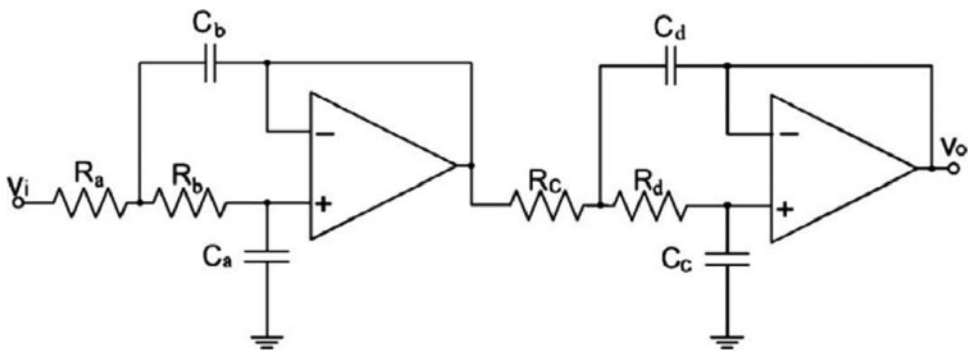

Figure 2. The fourth-order Butterworth filter.

factor of the first second-order Butterworth filter segment, and $H_2(s)$, ω_{cutoff2} and Q_{b2} are those of the second segment.

The transfer function of the Butterworth filter, as shown in Figure 2, is also given in the following in terms of the discrete circuit components.

$$H(s) = \frac{1}{s^2(R_aR_bC_cC_d) + s(R_aC_a + R_bC_a) + 1} \times \frac{1}{s^2(R_cR_dC_aC_b) + s(R_cC_c + R_dC_c) + 1} \quad (20)$$

Compare Equations (19) and (20), the expression for cut-off frequency and quality factors are obtained as

$$\omega_{\text{cut-off1}} = \frac{1}{\sqrt{R_aR_bC_aC_b}} \quad (21)$$

$$\omega_{\text{cut-off2}} = \frac{1}{\sqrt{R_cR_dC_cC_d}} \quad (22)$$

$$Q_{b1} = \frac{\sqrt{R_aR_bC_aC_b}}{R_aC_a + R_bC_a} \quad (23)$$

$$Q_{b2} = \frac{\sqrt{R_cR_dC_cC_d}}{R_cC_c + R_dC_c} \quad (24)$$

For a Butterworth filter, if the cost function error, $\cos tF_{\text{error}}$, is associated with the cost error of the cut-off frequency and the quality factor, named as $\cos tF_{\omega}$ and $\cos tF_{Q}$, respectively, then the total cost function error is given as Equations (25)–(27) (Sheta, 2010):

$$\cos tF_{\text{error}} = \frac{1}{2}(\cos tF_{\omega} + \cos tF_{Q}) \quad (25)$$

$$\cos tF_{\omega} = \frac{1}{\omega_c}(|\omega_{\text{cut-off1}} - \omega_c| + |\omega_{\text{cut-off2}} - \omega_c|) \quad (26)$$

$$\cos tF_{Q} = \left(\left|Q_{b1} - \frac{1}{0.7654}\right| + \left|Q_{b2} - \frac{1}{1.8478}\right|\right) \quad (27)$$

where the value of ω_c is fixed at 10 k rand/s.

The target of the Butterworth filter design is to minimise the total function error, with certain component ranges: the resistor values reside within the range of 103–106 Ω, and the capacitor values lie within the range of 10-9–10-6F (Bose, Biswas, & Vasilakos, 2014). The proposed algorithm, GAEA, runs to optimise the fourth-order Butterworth filter successfully for three times. The MIT are set as 2000, the population size of the iteration is set as 100. Three groups of solutions are shown in Table 10, comparing with the best solution in Bose's paper.

Consulting from Table 10, the cost error of the solution obtained from GAEA is about one-tenth of the best solution in Bose's paper. Unfortunately, The values of the circuit components in the three groups of solutions gotten by GAEA are different obviously, which means the Butterworth filter design problem is hard to solve for the existence of the local optima. As a whole, the results show that

Table 10. Values of the circuit components and the cost error of the Butterworth filter design.

	Values of the components								Total error
Algorithm	R_a (kΩ)	R_b (kΩ)	R_c (kΩ)	R_d (kΩ)	C_a (nF)	C_b (nF)	C_c (nF)	C_d (nF)	CostF_{error}
GAEA	5.21	4.45	4.17	1.00	7.93	54.47	35.75	67.10	5.38E-4
	2.80	10.67	4.03	9.20	5.68	58.91	13.97	19.32	4.35E-4
	11.90	12.01	3.53	2.80	3.20	21.85	29.20	34.67	3.42E-4
(Bose et al. 2014)	9.2	18.9	11.50	16.40	2.7	21	6.6	8	6.1E-3

the GAEA has the capacity to obtain a better Butterworth filter.

6. Conclusions

In this paper, an improved algorithm, which combines AEA with the multivariate Gaussian copula EDA, has been proposed. It possesses macro search ability of EDA as well as the advantages of gradient methods and random searching of AEA. GAEA performs well especially for the functions which contain strong correlations between variables, i.e. *Rosenbrock* and *Neumaier3*. The improved algorithm maintains the ability of escaping from local optima. The results of 22 test functions show that the GAEA is superior to ABC, DE, AEA and CMAES with regard to rate of success, convergence speed, average values and standard deviation of the best function value. Furthermore, the GAEA is compared with an ARFO algorithm with CEC2013 and used to solve a filter design problem. The best results obtained by the GAEA are superior to that of literature, which also demonstrates the effectiveness of the proposed algorithm

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work is supported by the National Natural Science Foundation of China [project number 21676086], [project number 21406064], [project number 21176072]; Natural Science Foundation of Shanghai [grant number 14ZR1410500]; Fundamental Research Funds for the Central Universities [grant number 222201717006].

Notes on contributors

Yihang Yang received BSc degree from East China University of Science and Technology (ECUST) in 2014. He is now a postgraduate in Control Science and Control Engineering in ECUST. His research direction is evolutionary computation.

Xiang Cheng received BSc degree from Changzhou Institute of Technology in 2013. He received master's degree in East China University of Science and Technology in 2016. His research direction was industrial process modelling and optimisation.

Junrui Cheng received BSc degree from Henan Industrial University in 2012. She received master's degree in East China University of Science and Technology in 2015. Her research direction was evolutionary computation.

Da Jiang received his PhD degree from Fudan University in 2009. He is now an associate professor in Department of Automation, East China University of Science and Technology. His researching interest includes industrial process modelling, optimisation and control, heat exchanger networks synthesis and process systems engineering.

Shaojun Li received his PhD degree from Dalian University of Technology in 2000. He is now a professor in Department of Automation, East China University of Science and Technology. His current research includes complex industrial process modelling, evolutionary computation and process systems engineering.

References

Ahn, C. W., An, J., & Yoo, J. C. (2012). Estimation of particle swarm distribution algorithms: Combining the benefits of PSO and EDAs. *Information Sciences, 192*, 109–119.

Ali, M. M., Khompatraporn, C., & Zabinsky, Z. B. (2005). A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization, 31*, 635–672.

Baluja, S., & Davies, S. (1997). *Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space*. Pittsburgh, PA: Carnegie-Mellon University.

Bose, D., Biswas, S., & Vasilakos, A. V. (2014). Optimal filter design using an improved artificial bee colony algorithm. *Information Sciences, 281*, 443–461.

Butterworth, S. (1930). On the theory of filter amplifiers. Experimental Wireless and the Wireless Engineer, 7, 536–541.

Cesar, R. M., Bengoetxea, E., Bloch, I., & Larranaga, P. (2005). Inexact graph matching for model-based recognition: Evaluation and comparison of optimization algorithms. *Pattern Recognition, 38*(11), 2099–2113.

Clemen, R. T., & Reilly, T. (1999). Correlations and copulas for decision and risk analysis. *Management Science, 45*, 208–224.

Deep, K., & Das, K. N. (2008). Quadratic approximation based hybrid genetic algorithm for function optimization. *Applied Mathematics and Computation, 203*, 86–98.

Dixon, W. J., & Mood, A. M. (1946). The statistical sign test. *Journal of the American Statistical Association, 41*, 557–566.

Gao, W. F., Liu, S. Y., & Huang, L. L. (2013). A novel artificial bee colony algorithm with Powell's method. *Applied Soft Computing, 13*, 3763–3775.

Gibbons, J. D., & Chakraborti, S. (2011). *Nonparametric statistical inference*. Berlin, Germany: Springer Berlin Heidelberg.

Hansen, N., Muller, S. D., & Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMAES). *Evolutionary Computation, 11*, 1–18.

Harth, E., & Tzanakou, E. (1974). Alopex: A stochastic method for determining visual receptive fields. *Vision Research, 14*, 1475–1482.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Ann Arbor, MI: The University of Michigan Press.

Inza, I., Larranaga, P., Etxeberria, R., & Sierra, B. (2000). Feature subset selection by Bayesian network-based optimization. *Artificial Intelligence, 123*, 108–115.

Kaeaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.

Kang, F., Li, J. J., & Li, H. J. (2013). Artificial bee colony algorithm and pattern search hybridized for global optimization. *Applied Soft Computing, 13*, 1781–1791.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization, 39*, 459–471.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks* (pp. 1942–1948). Piscataway, NJ: IEEE Press.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. (1983). Optimization by simulated annealing. *Science, 220*, 671–680.

Larranaga, P., & Lozano, J. A. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Boston, MA: Kluwer Academic Publisher.

Li, S. J., & Li, F. (2011). Alopex-based evolutionary algorithm and its application to reaction kinetic parameter estimation. *Computers & Industrial Engineering, 60*, 341–348.

Ma, L. B., Zhu, Y. L., Liu, Y., Tian, L. W., & Chen, H. N. (2015). A novel bionic algorithm inspired by plant root foraging behaviors. *Applied Soft Computing, 37*, 95–113.

Nelsen, R. B. (2007). *An introduction to copulas*. Berlin: Springer Science & Business Media.

Rao, R. V., Savsani, V. J., & Balic, J. (2012). Teaching-learning-based optimization algorithm for unconstrained and constrained real-parameter optimization problems. *Engineering Optimization, 44*, 1447–1462.

Sheta, A. F. (2010). Analogue filter design using differential evolution. *International Journal of Bio-Inspired Computation, 2*, 233–241.

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

Sun, J. Y., Zhang, Q. F., & Tsang, E. P. K. (2005). DE/EDA: A new evolutionary algorithm for global optimization. *Information Sciences, 169*, 249–262.

Wang, Y., Li, B., & Weise, T. (2010). Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems. *Information Sciences, 180*, 2405–2420.

Wang, Y. P., & Dang, C. Y. (2007). An evolutionary algorithm for global optimization based on level-set evolution and Latin squares. *IEEE Transactions on Evolutionary Computation, 11*, 579–595.

Yang, X. S., Karamanoglu, M., & He, X. S. (2014). Flower pollination algorithm: A novel approach for multi-objective optimization. *Engineering Optimization, 46*, 1222–1237.