Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Sciences 169 (2005) 249–262
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

DE/EDA: A new evolutionary algorithm for global optimization ☆

Jianyong Sun, Qingfu Zhang *, Edward P.K. Tsang

Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, UK

Received 1 August 2002; received in revised form 26 January 2004; accepted 27 June 2004

Abstract

Differential evolution (DE) was very successful in solving the global continuous optimization problem. It mainly uses the distance and direction information from the current population to guide its further search. Estimation of distribution algorithm (EDA) samples new solutions from a probability model which characterizes the distribution of promising solutions. This paper proposes a combination of DE and EDA (DE/EDA) for the global continuous optimization problem. DE/EDA combines global information extracted by EDA with differential information obtained by DE to create promising solutions. DE/EDA has been compared with the best version of the DE algorithm and an EDA on several commonly utilized test problems. Experimental results demonstrate that DE/EDA outperforms the DE algorithm and the EDA. The effect of the parameters of DE/EDA to its performance is investigated experimentally.

Keywords: Differential evolution; Estimation of distribution algorithm; Global continuous optimization problem

☆ This work was supported by EPSRC under Grant GR/R64742/01 and a Research Promotion Fund grant from the University of Essex.

* Corresponding author. Fax: +44 1206 872 788.

E-mail addresses: jysun@essex.ac.uk (J. Sun), qzhang@essex.ac.uk (Q. Zhang), edward@essex.ac.uk (E.P.K. Tsang).

0020-0255/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2004.06.009

1. Introduction

The global optimization problem arises in almost every field of science, engineering, and business. For example, engineers may want to design a car that provides the best performance. To achieve this, the engineers need to optimize the configuration parameters of the car. The finding of the best parameter configuration falls into the global optimization category. An enormous amount of efforts have been devoted to solving the global continuous optimization (e.g., see [1–3]). The major challenge of the global continuous optimization is that the problems to be optimized may have many local optima. Evolutionary algorithms (EA) [16,17] have been proposed for solving the global continuous optimization problem. EA analogizes the evolution process of biological population which can adapt the changing environments to the finding of the optimum of the optimization problem through evolving a population of candidate solutions.

Differential evolution (DE) [4,5] is one of the most successful EAs for the global continuous optimization problem. In fact, it was the best EA-like algorithm for the global continuous optimization problem in the first International Contest on Evolutionary Optimization (ICEO). DE extracts the differential information (i.e., distance and direction information) from the current population of solutions to guide its further search. However, DE has no mechanism to extract and use global information about the search space.

Estimation of distribution algorithm (EDA) [6–14] is a new class of EAs. EDA directly extracts the global statistical information about the search space from the search so far and builds a probability model of promising solutions. New solutions are sampled from the model thus built. Several EDAs [11–14] have been proposed for the global continuous optimization problem. These algorithms are very promising, but much work needs to be done to improve their performances.

An efficient evolutionary algorithm should make use of both the local information of solutions found so far and the global information about the search space. The local information of solutions found so far can be helpful for exploitation, while the global information can guide the search for exploring promising areas. The search in EDAs is mainly based on the global information, but DE on the distance and direction information which is a kind of local information. Therefore, it is worthwhile investigating whether combining DE with EDA could improve the performance of the DE algorithm and EDA.

This paper proposes an algorithm combining DE and EDA (DE/EDA). In the DE/EDA offspring generation scheme, part of a new solution is generated in the DE way, but the other part is sampled from a probability model. In such way, both the global information and local information are used to guide the further search. DE/EDA is compared with the best version of the DE algorithm [5] and the EDA on several commonly utilized test problems. Experimen-

tal results demonstrate that DE/EDA outperforms both the DE algorithm and the EDA. Furthermore, the effect of the parameters of the proposed algorithm is also experimentally studied in this paper.

The rest of the paper is organized as follows. Section 2 defines the global continuous optimization problem. In Section 3, the DE algorithm and EDA are briefly introduced. Section 4 describes the proposed DE/EDA. The experimental simulation results are reported in Section 5. Section 6 concludes the paper.

2. Problem definition

We consider the following unconstrained global continuous optimization problem:

$$\min f(x),$$

where $f(x)$ is a continuous real-value function defined on $X \subseteq R^n$ and it can be non-smooth. For constrained global continuous optimization problems, we can transform them into unconstrained ones. Therefore, we will focus on unconstrained global continuous optimization in this paper.

3. DE and EDA

3.1. Differential evolution

In this section, we briefly review the DE algorithm. The DE algorithm has several versions, we consider the best one proposed in [5]. This algorithm maintains a population of N points in every generation, where each point is a potential solution and N is a control parameter. The algorithm evolves and improves the population iteratively. In each generation, a new population is generated based on the current population. To generate offsprings for the new population, the algorithm extracts distance and direction information from the current population members and adds random deviation for diversity. If an offspring has a lower objective function value than a predetermined population member, it will replace this population member. This evolution process continues until a stopping criterion is met (e.g., the current best objective function value is smaller than a given value or the number of generations is equal to a given maximum value). In generation k, we denote the population members by $x_1^k, x_2^k, \ldots, x_N^k$. The DE algorithm is given as follows:

DE algorithm

Step 1: Set $k := 0$, and randomly generate N points $x_1^0, x_2^0, \ldots, x_N^0$ from X to form an initial population;

Step 2: For each point x_i^k $(1 \leqslant i \leqslant N)$, execute the DE offspring generation scheme to generate an offspring x_i^{k+1};
Step 3: If the given stop criteria are not met, set $k := k + 1$, goto Step 2.

The DE offspring generation scheme for generating x_i^{k+1} is given as follows:
DE offspring generation scheme

Step 1: Choose one point x_d randomly such that $f(x_d) \leqslant f(x_i^k)$, another two points x_b, x_c randomly from the current population and a subset $S = \{j_1, \ldots, j_m\}$ of the index set $\{1, \ldots, n\}$, while $m < n$ and all j_i mutually different;
Step 2: Generate a trial point $u = (u_1, u_2, \ldots, u_n)$ as follows:
Step 2.1: DE Mutation
Generate a temporary point z as follows:

$$z = (F + 0.5) \times x_d + (F - 0.5) \times x_i + F \times (x_b - x_c), \tag{1}$$

where F is a given control parameter;
Step 2.2: DE Crossover
For $j \in S$, u_j is chosen to be z_j; for $j \notin S$, u_j is chosen to be $(x_i^k)_j$;
Step 3: If $f(u) \leqslant f(x_i^k)$, set $x_i^{k+1} := u$; otherwise, set $x_i^{k+1} := x_i^k$.

Notice that Step 2.2 (DE Crossover) is actually implementing the well-known uniform crossover [16] between z and x_i^k. Therefore, m or S is determined by the probability of crossover which is a parameter of the DE algorithm.

It can be seen that the point z is generated by combing the current point x_i with a better point x_d which is randomly selected from the current population, and a randomly sampled vector differentials $(x_b - x_c)$. As argued in [5], the utilization of the sampled vector differentials have several advantages. Firstly, since the mean of the distribution of differentials is always zero, there is no sampling bias. This is helpful for preserving the population diversity. Secondly, the standard deviation of the differential's distribution could change along with the size and shape of the population in the search space which is valuable for problems whose parameters exhibit vastly different ranges and sensitivities. Thirdly, the scheme works well as a local optimizer since the differential in a converging population will eventually tend to zero. Especially, for the special case $n = 2$, $S = \{1,2\}$, $F = 1$ and $c = i$ (i is the index in Step 2.1 of the DE offspring generation scheme), the DE offspring generation scheme reduces to the reflection operator in the famous simplex optimization method [15] (only with a slight difference).

Due to its ability to maintain the diversity and to do local search, the DE algorithm performs better than some other EAs. But the DE algorithm has no mechanism to directly use the global information about the search space to guide the population towards promising areas.

3.2. Estimation of distribution algorithm

Let Pop(t) be the population of solutions at generation t. EDAs work in the following iterative way:

Step 1: Selection. Select M promising solutions from Pop(t) to form the parent set $Q(t)$ by a selection method (e.g., truncation selection);
Step 2: Modelling. Build a probabilistic model $p(x)$ based on the statistical information extracted from the solutions in $Q(t)$;
Step 3: Sampling. Sample new solutions according to the constructed probabilistic model $p(x)$;
Step 4: Replacement. Fully or partly replace solutions in Pop(t) by the sampled new solutions to form a new population Pop($t + 1$);

One of the major issues in EDAs is how to select parents. A widely-used selection method in EDA is the truncation selection. In the truncation selection, individuals are sorted according to their objective function values. Only the best individuals are selected as parents. The proposed DE/EDA employs the truncation selection.

Another major issue in EDAs is how to build a probability distribution model $p(x)$. In EDAs for the global continuous optimization problem, the probabilistic model $p(x)$ can be a Gaussian distribution [11], a Gaussian mixture [12,13], a histogram [14], or a Gaussian model with diagonal covariance matrix (GM/DCM) [12].

GM/DCM is used in our algorithm. In GM/DCM, the joint density function of the k-th generation is written as follows:

$$p_k(x) = \prod_{i=1}^{n} \mathcal{N}(x_i; \mu_i^k, \sigma_i^k), \tag{2}$$

where

$$\mathcal{N}(x_i; \mu_i^k, \sigma_i^k) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{\frac{1}{2}\left(\frac{x_i-\mu_i}{\sigma_i}\right)^2}.$$

In (2), the n-dimensional joint probability distribution is factorized as a product of n univariate and independent normal distributions. There are two parameters for each variable required to be estimated in the k-th generation: the mean, μ_i^k, and the standard deviation, σ_i^k. They can be estimated as follows:

$$\hat{\mu}_i^k = \overline{X}_i^k = \frac{1}{M}\sum_{t=1}^{M} x_{t,i}^k, \qquad \hat{\sigma}_i^k = \sqrt{\frac{1}{M}\sum_{t=1}^{M}(x_{t,i}^k - \overline{X}_i^k)^2}, \tag{3}$$

where $(x_{1,i}^k, x_{2,i}^k, \ldots, x_{M,i}^k)$ are values of the i-th variable of the selected M parent solutions in the k-th generation.

4. DE/EDA algorithm

The most important operation in the DE algorithm is to generate offspring. As we have seen from the above DE offspring generation scheme, each offspring is generated by crossing a solution from the current population and a solution obtained by the DE mutation.

Inspired by EDA, which try to guide its search towards a promising area by sampling new solutions from a probability model, we incorporate the EDA mechanism into the DE algorithm in order to create solutions which are more promising than solutions generated by the DE recombination (crossover and mutation), and consequently, to explore the search space more effectively. At generation k, the proposed DE/EDA offspring generation scheme works as follows:

DE/EDA offspring generation scheme

Step 1: Select the best M solutions from the current population, construct a probability model according to (3):

$$p_k(x) = \prod_{i=1}^{n} \mathcal{N}(x_i; \hat{\mu}_i^k, \hat{\sigma}_i^k).$$

Step 2: Generate a trial solution $u = (u_1, u_2, \ldots, u_n)$ as follows: for all $j = 1, 2, \ldots, n$

if $(rand() < \delta)$

$$u_j = [(x_i^k)_j + (x_d)_j]/2 + F \times [(x_d)_j - (x_i^k)_j + (x_b)_j - (x_c)_j]; \tag{4}$$

else

u_j is sampled according to $\mathcal{N}(x_i; \hat{\mu}_j^k, \hat{\sigma}_j^k)$,

where $rand()$ is an uniform random number in $(0, 1)$, and δ $(0 \leqslant \delta \leqslant 1)$ is a parameter.

Step 3: If $f(u) < f(x_i^k)$, set $x_i^{k+1} := u$; otherwise, set $x_i^{k+1} := x_i^k$.

The above offspring generation scheme is similar to the DE crossover. Like the DE offspring generation scheme, one part of a trial solution generated comes from the DE mutation (3). But the other part of the trial solution is sampled in the search space from the constructed probability distribution model. Therefore, a trail solution generated by the DE/EDA offspring generation scheme is based on the differential information and global statistical information. δ is used to balance contributions of the global information and the differential information.

4.1. DE/DEA algorithm

In the following, we incorporate the DE/EDA offspring generation scheme into the DE algorithm. The DE/EDA algorithm is given as follows:
DE/EDA algorithm

Step 1: Randomly generate N solutions $x_1^0, x_2^0, \ldots, x_N^0$ from the feasible search space to form an initial population, set $k := 0$;
Step 2: Generate a new solution x_i^k according to the DE/EDA offspring generation scheme described in the above section;
Step 3: If the given stopping criterion is not met, $k := k + 1$, goto Step 2.

Parameters involved in the proposed algorithm are N, δ and F. The number of solutions selected in Step 1 of the DE/EDA offspring generation scheme $M = N/2$. One possible stopping criterion is to stop when the current objective function value is smaller than a given value or the number of objective function evaluations is equal to a given maximal value.

Compared with the DE algorithm, DE/EDA has only a small extra computational cost in constructing the probability model. On the other hand, DE/EDA has the ability to utilize the global statistical information collected from the previous search, and it also can use the differential information in DE way.

5. Experimental results

In this section, DE/EDA was applied to several commonly utilized minimization problems in order to verify its performance and compared with the best version of DE algorithm [5] and an EDA. In fact, the EDA to be compared is just DE/EDA with $\delta = 0.0$. Furthermore, the effect of the parameters $\delta \in [0.0, 1.0]$ and $F \in [-1.0, 1.0]$ to the performance of DE/EDA is experimentally investigated.

In order to make a fair comparison, we perform 20 times of independent runs on each problem for these algorithms to be compared within given objective function evaluations. Tables 1 and 2 summarize the results of the comparison for problems with dimension $D = 5$ and $D = 10$, respectively. Since the DE algorithm was executed to these test problems in [5], Tables 1 and 2 also include all of the available results from [5]. The test problems are Generalized Rosenbrock, Modified Langerman, Shekel's Foxholes, Epistatic Michalewicz, and Tchebychev Polynomial, denoted by G.Ros., M.Lan., S.Fox., E.Mic. and T.Pol. in the tables, respectively. In Tables 1 and 2, denote by BVAT the best value found during the 20 trials, by ENES the total number of function evaluations over the 20 trials, divided by the number of trials that successfully attain the VTR (value-to-reach), by $\bar{f}$ the average best objective function

Table 1
The comparison results ($D = 5$)

Fun.	VTR	DE			EDA			DE/EDA		
		$\bar{f}$	BVAT	ENES	$\bar{f}$	BVAT	ENES	$\bar{f}$	*BVAT*	ENES
G.Ros.	1.0e−6	VTR	0.0	4561	2.442	1.392	N/A	VTR	0.0	4554
M.Lan.	N/A	−0.963	−0.965	N/A	−0.899	−0.947	N/A	−0.964	−0.965	N/A
S. Fox.	N/A	−10.20	−10.40	N/A	−6.342	−8.439	N/A	−10.40	−10.40	N/A
E.Mic.	−4.687	−4.67	−4.826	5339	−3.745	−4.628	N/A	−4.680	−4.687	2894
T.Pol.	1.0e−7	VTR	4.3e−29	17325	5.24e+2	2.69e+2	N/A	VTR	0.0	14009

Table 2
The comparison results ($D = 10$)

Fun.	VTR	DE			EDA			DE/EDA		
		$\bar{f}$	BVAT	ENES	$\bar{f}$	BVAT	ENES	$\bar{f}$	BVAT	ENES
G.Ros.	1.0e−6	VTR	0.0	27169	8.168	7.755	N/A	VTR	0.0	22709
M.Lan.	N/A	−0.8343	−0.965	N/A	−0.180	−0.617	N/A	−0.902	−0.965	N/A
S.Fox.	N/A	−10.20	−10.28	N/A	−0.900	−1.446	N/A	−10.26	−10.28	N/A
E.Mic.	−9.660	−9.44	−9.660	77365	−7.166	−9.184	N/A	−9.450	−9.660	57892
T.Pol.	1.0e−7	VTR	5.7e−29	78937	3.39e+6	1.69e+6	N/A	VTR	0.0	71293

values found in 20 trials. The algorithm stops when the given VTR is reached or the objective function evaluations exceed the given maximal number of objective function evaluations. "N/A" means not applicable. For the Modified Langerman and the Shekel's Foxholes, since no VTRs are available, no ENES values are given in Tables 1 and 2. Because the EDA can not reach the VTRs for all the problems, no ENES values are given in the EDA column in the tables. Table 3 lists the parameter settings of DE/EDA for the test problems.

Figs. 1 and 2 present the solution qualities and computational costs of DE/EDA, the DE algorithm and the EDA on these test problems with dimension $D = 5$ and $D = 10$, respectively. The evolution procedures of the test problems from G.Ros. to T.Pol. with dimensions 5 and 10 are labelled from (a) to (e) in Figs. 1 and 2, respectively. From Tables 1 and 2, it can be seen that the DE/EDA and DE gives about the same best solutions except the problem Epistatic Michalewicz with dimension $n = 5$, where the solution found by DE is slightly better than that of DE/EDA. But DE/EDA needs less function evaluations to reach VTR and it can obtain smaller average objective function value $\bar{f}$. The EDA is the worst in terms of the solution quality and computational cost.

Figs. 1 and 2, except Fig. 1(c), show that the average best objective function values found in DE/EDA decrease faster than these found in the DE algorithm. From Fig. 1(c), it can be seen that after approximate 1300 generations, the average objective function value obtained by DE/EDA is better than the value found by the DE algorithm for Epistatic Michalewicz problem with dimension $n = 5$.

Table 3
Parameter settings of DE/EDA

Function	D	*Parameter settings*		
		N	F	δ
G.Ros.	5	20	0.6	0.9
	10	40	0.6	0.9
M.Lan.	5	100	0.4	0.3
	10	150	0.5	0.8
S.Fox.	5	150	−0.5	0.9
	10	150	−0.5	0.9
E.Mic.	5	40	0.5	0.7
	10	40	0.4	0.7
T.Pol.	9	81	0.5	0.9
	17	170	0.5	0.9

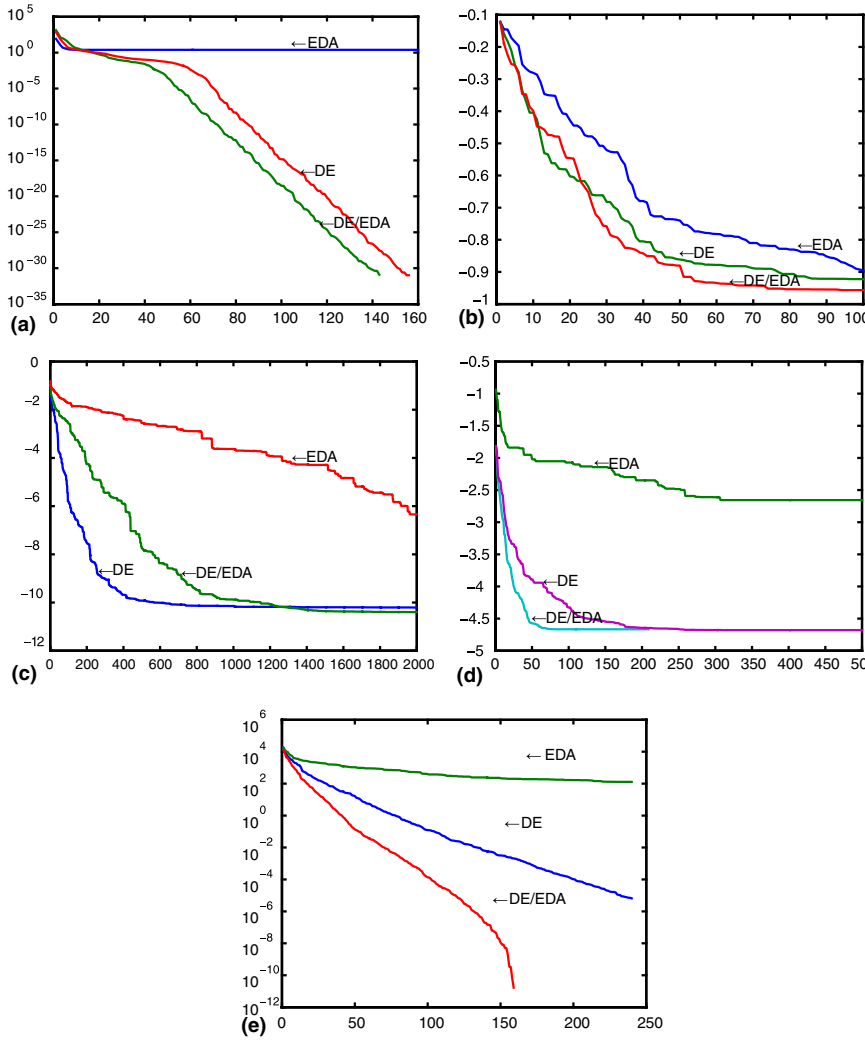

Fig. 1. The evolution procedure for solving problems with dimension 5. (For colour see online version.)

In summary, we can claim that DE/EDA performs better than the DE algorithm and the EDA in terms of solution quality within given objective function evaluations for the test problems.

Furthermore, the effect of the parameter δ of DE/EDA is firstly investigated in case that F is fixed. To make a fair comparison of the effect of different δ's to the performance of DE/EDA, DE/EDA terminates when the VTR value is reached or the average objective function value of the population has not chan-

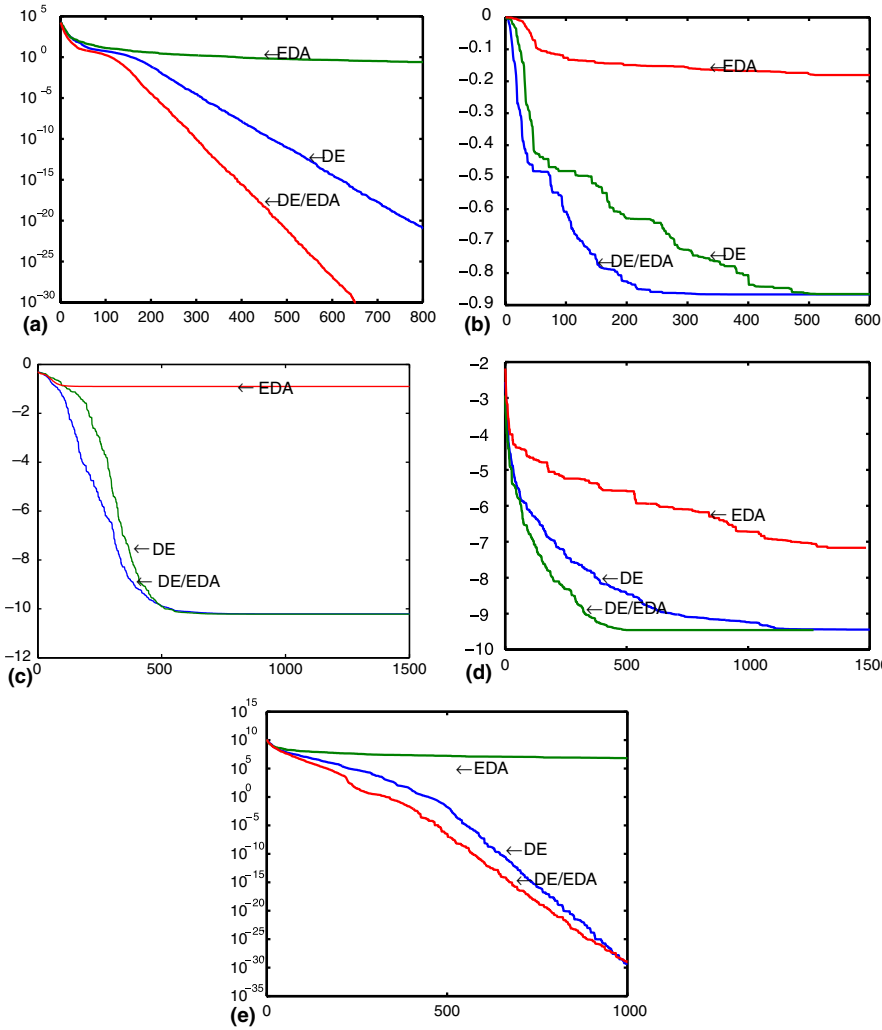

Fig. 2. The evolution procedure for solving problems with dimension 10. (For colour see online version.)

ged much (less than 10^{-3}) within consecutive 100 generations. The average objective function value $\bar{f}$ and the number of objective function evaluations (nfe) when the algorithm stops are compared and listed in Table 4.

From Table 4, we can see that as δ increases, the performance of DE/EDA firstly becomes better until the best performance is achieved. Then the performance of the algorithm becomes worse. The turning point depends on the problem to be optimized: for the Modified Langerman with dimension

Table 4
The effect of δ to the performance of DE/EDA

δ	S.Fox. ($D = 10$)		G.Ros. ($D = 5$)		G.Ros. ($D = 10$)		M.Lan. ($D = 10$)	
	$\bar{f}$	*nfe*	$\bar{f}$	*nfe*	$\bar{f}$	*nfe*	$\bar{f}$	*nfe*
0.0	−1.219	29160	2.669	3600	7.801	7892	−0.255	39315
0.1	−1.477	25875	2.560	3960	7.773	9928	−0.341	43635
0.2	−1.477	26745	2.348	4700	7.224	15536	−0.452	44985
0.3	−1.477	27420	1.671	8112	3.992	82300	−0.688	58350
0.4	−1.477	28920	1.074	14206	2.132	82470	−0.711	55155
0.5	−1.474	36405	0.957	12782	0.389	79428	−0.638	54675
0.6	−1.475	45015	0.248	11534	0.024	57008	−0.882	62010
0.7	−1.471	61410	0.053	9214	8.36e−04	29120	−0.888	61695
0.8	−7.608	152355	3.93e−03	4958	1.21e−04	23328	−0.902	50295
0.9	−10.26	114060	6.87e−07	4118	1.48e−05	19852	−0.609	38280
1.0	−9.488	100275	1.50e−03	5762	1.930	30672	−0.568	30360

10, $\delta = 0.8$; for the Shekel's Foxholes with dimension 10 and the Generalized Rosenbrock problems, $\delta = 0.9$. It can also be seen that with an inappropriate δ, DE/EDA will be stuck in a local minima within a few objective function evaluations (e.g., $\delta = 0.1$ in Shekel's Foxholes with dimension 10, nfe = 25875). In other words, an inappropriate δ will cause the pre-mature convergence of DE/EDA.

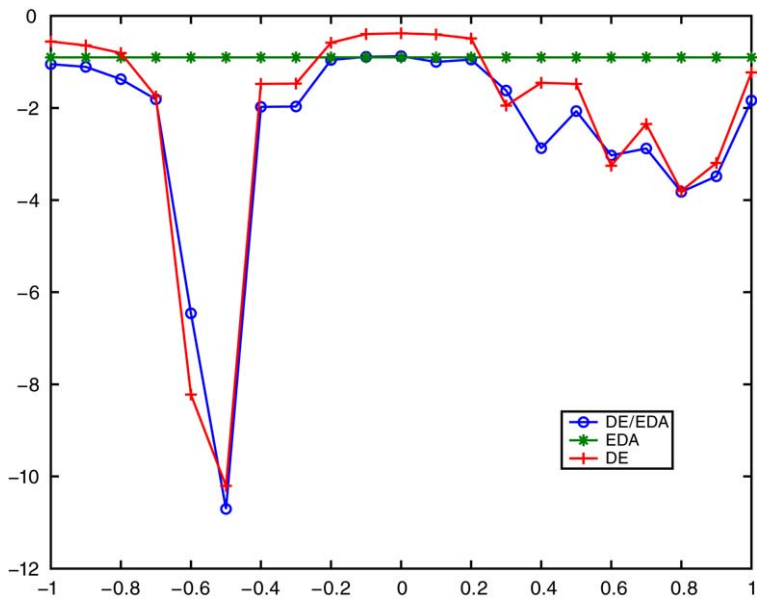

Fig. 3. The effect of F to the performance of DE/EDA taking the Shekel's Foxholes as an example. (For colour see online version.)

The effect of F with fixed δ is explored as well. Fig. 3 shows the effect of $F \in [-1.0, 1.0]$ with $\delta = 0.9$ taking the Shekel's Foxholes with dimension 10 as an example. From Fig. 3, it can be seen that the combination of global information obtained by the EDA can improve the performance of the DE algorithm: for most F values, the performance of DE/EDA is better than that of the DE algorithm. The local information from the DE algorithm is of benefit to the performance of the EDA as well: in the case of $\delta = 0.9$, the average objective function value found by DE/EDA is better than the value found by the EDA ($\delta = 0.0$) for most F's. Therefore, we can claim that the combination of local information and global information can indeed improve the performance of evolutionary algorithm.

6. Conclusion

In this paper, we have proposed an improved DE algorithm based on the estimation of distribution algorithm, called DE/EDA. In DE/EDA, new promising solution is created by DE/EDA offspring generation scheme, in which local information (obtained by the DE mutation) and global information (extracted from a population of solutions by the EDA modelling) are incorporated together.

We executed DE/EDA to solve several commonly utilized test problems with different dimensions and compared with the best version of the DE algorithm described in [5] and the EDA. The experimental results demonstrated that for the test problems, DE/EDA outperforms the DE algorithm and the EDA in terms of solution quality within given objective function evaluations.

The effect of the parameter δ with fixed F to the performance of DE/EDA is experimentally investigated. The computational results show that in case that F is fixed, as δ increases, DE/EDA firstly performs better until the best performance is achieved, then the performance becomes worse. The value of δ for the best performance depends on the problem to be optimized. The effect of F is explored as well with fixed value of δ. From the experiments, we can claim that the combination of local information and global information can improve the performance of evolutionary algorithm.

In the future, we will combine EDAs with other evolutionary algorithms for solving hard combinatorial optimization problems.

References

[1] A. Torn, A. Zilinskas, Global Optimization, Springer-Verlag, Berlin, 1989.
[2] W.W. Hager, D.W. Hearn, P.M. Pardalos (Eds.), Large Scale Optimzation: State of the Art, Kluwer Academic Publishers, 1994.

[3] C.A. Floudas, P.M. Pardalos (Eds.), State of the Art in Global Optimization: Computational Methods and Applications, Kluwer Academic Publishers, 1996.
[4] R. Storn, K. Price, Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces, Technical Report TR-95-012, International Computer Science Instititute, Berkely, California, 1995.
[5] K. Price, Differential Evolution vs. the Functions of the 2nd ICEO, In: Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97), Indianapolis, USA, 13–16 April 1997, pp. 153–157.
[6] H. Mühlenbein, The equation for response to selection and its use for prediction, Evolutionary Computation 5 (3) (1998) 303–346.
[7] B.T. Zhang, A bayesian framework for evolutionary computation, In: Proceedings of the 1999 Congress on Evolutionary Computation 1 (1999) 722–728.
[8] A. Ochoa, EBBA—an evolutionary best-basis algorithm. In: Proceedings of the Second Symposium on Artificial Intelligence, ISAS' 99, 1999, pp. 93–98.
[9] A. Ochoa, H. Mühlenbein, M. Soto, A factorized distribution algorithm using single connected bayesian networks, PPSN (2000) 787–796.
[10] M. Pelikan, D.E. Goldberg, E. Cantú-Paz, BOA: The Bayesian optimization algorithm, In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO99), I, 1999, pp. 525–532. Also IlliGAL Report No. 99003.
[11] S. Rudlof, M. Koppen, Stochastic Hill-Climbing with Learning by Vectors of Normal Distributions, Nagoya, Japan, 1996.
[12] P. Larrañaga, J.A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publishers, sdsdsd, 2001.
[13] P.A.N. Bosman, D. Thierens, Expanding from Discrete to Continuous EDAs: The IDEA, In: Proceedings of Parallel Problem Solving from Nature, PPSN-VI, 2000, pp. 767–776.
[14] S. Tsutsui, M. Pelikan, D.E. Goldberg, Evolutionary Algorithm Using Marginal Histogram Models in Continuous Domain, In Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop, 2001, pp.230–233, San Francisco, CA.
[15] R. Fletcher, Practical Methods of Optimization, Vol. 1, John Wiley and Sons, New York, 1980.
[16] L. Davis, Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.
[17] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.