

An Improved Differential Evolution Algorithm and its Application in Reaction Kinetic Parameters Estimation

Dan Xu Shaojun Li Feng Qian

Institute of Automation, East China University of Science and Technology, Shanghai 200237, P. R. China

Abstract

Differential evolution algorithm (DE) is a simple efficient optimization technique, but it is easily trapped in the local optima. This paper presents an improved differential algorithm (IDE) based on Alopex (Algorithms of Pattern Extraction) where “noise” strategy according to the learning experience and memory selection are used. In order to seek for better result Alopex operator contracts searching area self-adaptively during iteration process. The performance of IDE is tested by several benchmark functions. Results show that the IDE algorithm overcomes the disadvantages of the original DE and possesses higher precision. Finally IDE is successfully applied to reaction kinetic parameters estimation.

Keywords: Differential Evolution Algorithm, Alopex, Parameters Estimation

1. Introduction

Many practical problems are non-differentiable, non-linear, multi-dimensional noisy, flat, or have many local optima, constraints or randomness. The traditional gradient-based algorithms are notorious for trapping into local optima. However, DE algorithm [1], proposed by Storn and Price, which is a stochastic, population-based for global optimization over continuous spaces, has become a new study focus. It is a simple and surprisingly efficient algorithm which is a stochastic, population-based for global optimization over continuous spaces. Its effectiveness and efficiency have been successfully demonstrated in many application fields such as pattern recognition [1], communication [2], mechanical engineering [3] and so on.

Parameters and learning strategies involved in DE are highly dependent on the problems under consideration. Compared to other optimizing algorithms, such as Particle Swarm Optimization (PSO), Genetic Algorithm (GA) and Simulate Anneal (SA), DE is much faster, easier to be programmed and has comparative global searching ability. However, it has lower precision and not easily gets into

convergence. This motivates us to develop an improved algorithm to solve general problems more efficiently.

In the proposed IDE algorithm, the Alopex [4] operator and natural immune memory were used. In Alopex, “noise” strategy which redounds to improve local searching ability was proposed; self-adaptive step length makes searching process go forwards to global optimum; immune memory selection help get candidate solution which has better fitness value, self organizing and highly distribution [5]. To sum up IDE has advantages of combining global searching and local searching.

The reminder of the paper is organized as follows: In section 2, DE and Alopex are briefly described. The proposed IDE algorithm and implementation are provided in section 3. Then Benchmark functions are used to measure the performance of IDE, The results are presented in section 4. After that in section 5, IDE is applied to reaction kinetic parameters estimation. Finally, Section 6 concludes the paper.

2. A brief description of DE and Alopex

2.1. DE algorithm

DE algorithm is a population-based [6]-[7] search type algorithm. To start DE, we generate N_p design vectors randomly as the initial population, namely $[x_1^0, x_2^0, \dots, x_{N_p}^0]$. The size of population, N_p , is held constant throughout the optimization process. The dimension of each vector was determined by the problem which requires to be optimized.

To generate the new population, the algorithm extracts distance and direction information from the current population members and adds random deviation for diversity. Considering x_i^k as the target point in k^{th} iteration, a related mutation vector is $V_i^k = \{x_{i1}^k, x_{i2}^k, \dots, x_{iN_p}^k\}$, generally, there are several typical strategies in mutation phase [8] such as:

$$\text{“DE/best/1”}: V_i^{k+1} = x_{\text{best}}^k + F \square (x_{i1}^k - x_{i2}^k) \quad (1)$$

$$\text{"DE/rand/1"}: V_i^{k+1} = x_{i1}^k + F(x_{i2}^k - x_{i3}^k) \quad (2)$$

“DE/rand to best/1”:

$$V_i^{k+1} = x_i^k + F(x_{\text{best}}^k - x_i^k) + F(x_{i1}^k - x_{i2}^k) \quad (3)$$

“DE/best/2”:

$$V_i^{k+1} = x_{\text{best}}^k + F(x_{i1}^k - x_{i2}^k) + F(x_{i3}^k - x_{i4}^k) \quad (4)$$

“DE/rand/2”:

$$V_i^{k+1} = x_{i1}^k + F(x_{i2}^k - x_{i3}^k) + F(x_{i4}^k - x_{i5}^k) \quad (5)$$

Where F is a scaling factor (known as the mutation constant) in $[0,2]$, which controls the amplification of the difference between two individuals, so as to avoid the stagnation of evolving procedure [9], “ $i1, i2, i3, i4, i5$ ” represent the index of individuals selected randomly, “ k ” is the generation index and x_{best}^k is the best member of the previous generation.

The strategy highly depends on the problems under consideration, meanwhile it also determines the performance of optimization [10]-[11]. In this article, “DE/best/1” (strategy1), “DE/rand/1” (strategy 2) and “DE/rand to best/1” (strategy3) were used to test the performance of benchmark function in section 4.

In order to increase the diversity among the mutant parameter vectors, crossover is introduced. The trial vector is $U_{ji}^k = \{U_{1i}^k, U_{2i}^k, \dots, U_{ni}^k\}$ and the binomial crossover operation can be described by (6),

$$U_{ji}^k = \begin{cases} V_{ji}^k, & \text{if } (\text{rand}_j \leq CR) \\ x_{ji}^k, & \text{otherwise} \end{cases} \quad (j = 1, 2, \dots, n) \quad (6)$$

After that, in order to determine whichever V_{ji}^k or x_{ji}^k is transferred into the next generation, the objective function value of the two, namely $f(V_{ji}^k)$ and $f(x_{ji}^k)$, are compared and if an offspring has a lower objective function value than a predetermined population member, the new individual replaces the old one, as in (7),

$$x_i^{k+1} = \begin{cases} U_i^k, & \text{if } (f(U_i^k) < f(x_i^k)) \\ x_i^k, & \text{otherwise} \end{cases} \quad (i = 1, 2, \dots, N_p) \quad (7)$$

This evolving procedure continues until a stopping criterion is met.

2.2. The Alopex algorithm

Alopex has randomness which can get rid of local optima and find the best solution, meanwhile it has heuristic search from the infection of objective function which depends on changes of last independent variables. So this algorithm could find global optimum gradually.

A practical optimized problem can be summed up a problem which seeking extremum of objective function $E(x_1, x_2, \dots, x_n)$, where (x_1, x_2, \dots, x_n) are independent variables. The optimization strategy [12] of basal Alopex is:

$$x_i(t+1) = x_i(t) + \delta_i(t+1) \quad (8)$$

Where $\delta_i(t)$ is learning-rate (also known as the step size of independent variable) at the time of t^{th} .

$$\delta_i(t+1) = \begin{cases} -\delta_i(t) & \text{with probability } p_i(t) \\ +\delta_i(t) & \text{with probability } 1 - p_i(t) \end{cases} \quad (9)$$

$$p_i(t) = \frac{1}{1 + \exp(\mp c_i(t)/T)} \quad (10)$$

Where $p_i(t)$ is the direction choosing probability of variable $x_i(t)$ at the time of t , “+” indicates seeking maximum, “-” means seeking minimum, T is annealing “temperature” parameter.

$$\begin{aligned} c_i(t) &= [x_i(t-1) - x_i(t-2)][E(x_i(t-1)) - E(x_i(t-2))] \\ &= \Delta x_i(t) \Delta E(t) \end{aligned} \quad (11)$$

Where $\Delta x_i(t)$ is the difference between independent variable $x_i(t-1)$ and $x_i(t-2)$; meanwhile $\Delta E(t)$ is the difference between the values of objective function.

At each iteration, the algorithm makes a choice of direction randomly (has a forward direction or reverse direction). $p_i(t)$ depends on the correlation between $\Delta x_i(t)$ and $\Delta E(t)$, as given by (10).

T is the important parameter in optimization which affects the performance of the algorithm mostly. At the beginning of the algorithm, T is large to ensure “grabbling ability”, and then gradually reduces along with the searching. As in [13], this paper adopts a method in which T parameter updated every N iterations (for a suitably chosen N), the following annealing schedule refer to (12),

$$T = \frac{1}{Nm} \sum_{i=1}^m \sum_{t'=t-N}^t |c_i(t')| = \frac{\eta}{N} \sum_{t'=t-N}^t |\Delta E(t')| \quad (12)$$

From the annealing schedule given by (12), it is clear that T has corresponding decrease along with the reduction of the $\Delta E(t')$.

3. Implementation of IDE algorithm

In Alopex, $\delta_i(t)$ has important influence over algorithm’s performance. Usually, it needs to be decreased gradually, so as to not only keep global searching, but also actualize local search in corresponding extension for precision. The experiments show that if $\delta_i(t)$ is too small, the convergence rate can’t be guaranteed; on the other hand if $\delta_i(t)$ is big

enough, it may be miss the optimum closer current solution. The expression is given by (13),

$$\delta_i(t) = \frac{1}{20} \times \Delta x_i(t) \quad (13)$$

Immune system is seen as a complex of cells, organs that protect organisms against infections. In IDE algorithm, immune memory selection was added, the objective function corresponds to antigen; candidate solution of problems denotes antibody; Fitness value means affinity which depends on comparability of problems' solution; Alopex operation is similar to stimulation of antibodies which is good for improving the searching efficiency around the best solution. Memory is constructed of candidate solutions which have lower fitness value. When the searching stagnates in local optimum, Alopex generate reverse searching for the sake of getting rid of local optimum.

For convenience, we introduce the operating procedure of IDE algorithm using strategy 2.

Step1: Set iter=0, randomly generate Np points $x_1^0, x_2^0 \dots x_{N_p}^0$ as the initial population and select NM (generally NM=0.3Np) individuals randomly as memory variables, meanwhile, set the first point as the best one namely Gbest;

Step2: Calculate the fitness value, iter=iter+1;

Step3: For each point x_i^k ($1 \leq i \leq N_p$), select three individuals randomly, generate an mutant vector V_i^{k+1} by (2),

Step4: Suppose x_{i1}^k is an independent variable value at iterations (t-2), compute the corresponding objective function value $f(x_{i1}^k)$;

Step5: Generate a random number, $rand \in (0,1)$, when $rand \leq CR$, execute binomial crossover operation by (6);

Step6: The offspring x_i^{k+1} was generated by (7);

Step7: Suppose x_{i1}^{k+1} is an independent variable value at the time of (t-1), $f(x_{i1}^{k+1})$ is the corresponding objective function value, then compute the differences between x_{i1}^{k+1} and x_{i1}^k (namely $\Delta x_i(t)$) and difference of two objective

functions (namely $\Delta E(t)$) at the time of (t-1) and (t-2) respectively;

Step8: According to (10), compute the possibility of choice of direction, Update the independent variables by (9), (13) and (8);

Step9: Add memory individuals to new population, then calculate their objective function value. After that select the former NM candidate solutions construct new memory, thereinto the best one as Gbest, re-update the population using the former Np individuals;

Step10: If a stopping criterion (iter=Niter, or $G_{best}^{k+1} - G_{best}^k \leq 10^{-5}$) is met, stop the algorithm and output the result, otherwise turn to step 2.

By introducing Alopex operator and memory selection technique, the IDE algorithm is endowed with self-learning and exploiting ability which can get rid of local optimum in a great extend, Meanwhile it has no need for the objective function to be differentiable, and has a high potential for parallelism.

4. Results of Test Functions and Discussions

Benchmark functions were used to measure the performance and analyze the influence of differential strategy involved in DE and IDE. These test functions are Schaffers, Grievank, Sphere and Ackly whose expressions and parameters are demonstrated in the following table1 [14]-[15].

In DE and IDE algorithms, same parameters used were population size Np=50, scaling factor F=0.6, crossover rate CR=0.3, N=100 and iteration number Niter=2000. Table 2 lists the statistic results of performances comparison of DE and IDE for 30 times. P_{opt} denotes the probability of finding global optimum, S_{min} indicates the minimal generation of finding global minimal and S_{avg} means the average generation of finding global minimum.

Function	Expressions	Dim	Bounds	Optima
Schaffers	$f(x) = 0.5 - \frac{(\sin \sqrt{x_1^2 + x_2^2})^2 - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$	2	[-100, 100]	1
Grievank	$f(x) = \frac{1}{4000} \sum_{i=1}^n (x_i)^2 - \prod_{i=1}^n \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[-600, 600]	0
Sphere	$f(x) = \sum_{i=1}^n x_i^2$	30	[-100,100]	0
Ackly	$f(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)\right) + 20 + e$	30	[-32,32]	0

Table 1: The definition of benchmark test function.

The four functions' convergence maps of the IDE and DE algorithm are illustrated in figures 1-4, respectively. Abscissa axis denotes the evolutionary generation of algorithms; Vertical axis represents the

fitness of test function. For clarity, logarithmic format was used in vertical axis.

Generally speaking, the strategy varies with respect to the number of other random individuals that

are used to construct a new trial vector, as well as whether or not the current individual or the global best individual would be used. Meanwhile the strategy determines the performance of optimization mostly.

Function	Strategy	DE			IDE		
		P_{opt}	S_{min}	$S_{avg.}$	P_{opt}	S_{min}	$S_{avg.}$
Schaffers	1	63%	300	415	76%	120	322
	2	100%	970	1424	100%	280	847
	3	0	N/A	N/A	3%	860	860
Grievank	1	70%	770	815	74%	540	561
	2	100%	1750	1846	100%	1040	1080
	3	100%	710	917	100%	690	828
Sphere	1	100%	520	534	100%	370	387
	2	100%	1180	1212	100%	730	770
	3	100%	470	481	100%	450	467
Ackly	1	100%	830	844	100%	540	555
	2	100%	1870	1897	100%	1020	1076
	3	100%	730	750	100%	540	545

Table 2: The performances comparison of DE and IDE algorithms.

The experimental results show that strategy 1 has faster convergence rate than strategy 2 and strategy 3 mostly, nearly about half or even less than other strategies, inasmuch as strategy 1 joins the current best member to construct a new trial vector which leads to get higher convergence rate. But for some functions, its astringency can't be fully guaranteed, e.g. Schaffers and Grievank functions, the probability of finding global minimal about 60%-70%. However, strategy 2 has the advantage of finding global optimum nearly all the time. We explain that strategy 2 constructs new vector of 3 random individuals which maintains good diversity for a long time [16]. Compared with Strategy 1 and 2, generally, strategy 3 seems to combine the characters of two strategies with average convergence rate and ordinary capability. However it's not changeless, take Schaffers function for instance, it can't find any global optimum solution at all over 30 runs for strategy 3.

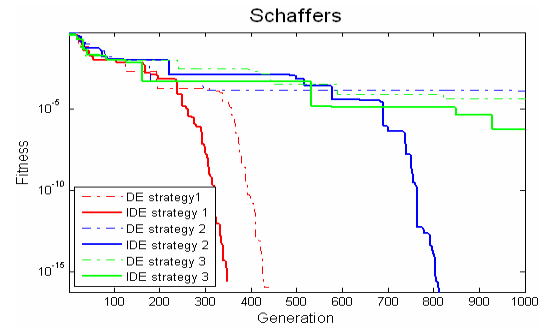


Fig. 1: Convergence Graph for Schaffers.

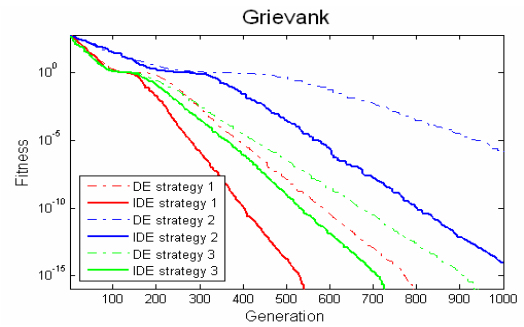


Fig. 2: Convergence Graph for Grievank.

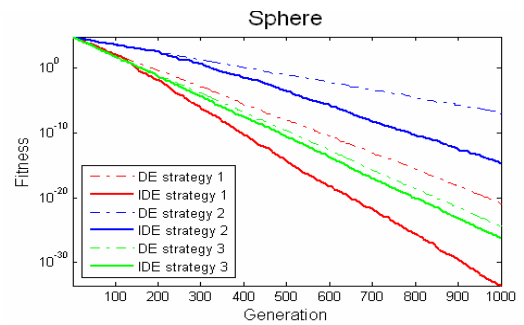


Fig. 3: Convergence Graph for Sphere.

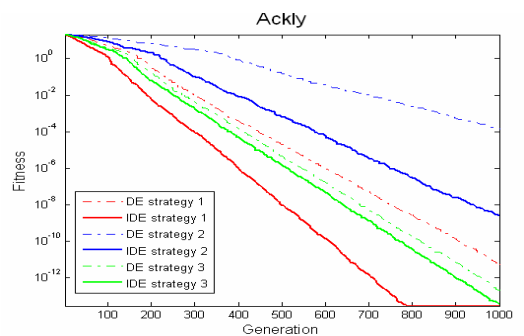


Fig. 4: Convergence Graph for Ackly.

When compared IDE with DE, it is clear that the IDE has better performance of global convergence, namely larger probability and fewer average generations of finding global optima using three differential strategies. Concretely, take strategy 1 for example, for Schaffers function, DE needs 415

generations to find global optimum, however IDE only requires 322 average generations, meanwhile the percentage of finding global optimum increased from 63% to 76%. Similarly, the IDE also shows its superiority of Grievank Sphere and Ackly functions. It indicates that the IDE improves the speed and accuracy of convergence, especially for those functions which have multi-modality. Therefore IDE algorithm is more effective than original DE algorithm.

5. Application of Kinetic Parameters Estimation

As the disadvantage of the existing transaction technique of the wastewater, a new high efficient technique supercritical water oxidation which can degrade organic compound thoroughly without deleterious byproducts becomes one of the most popular reaction processes.

Kinetic parameters estimation in Supercritical Water Oxidation is a problem with high dimension, high non-linearity besides possess many local extreme points. Up to now, several methods came into use. Literatures [17] elaborated the result of an investigation into the global kinetics of and reaction products from 2-chlorophenol (2CP) oxidation in supercritical water oxidation using the way of nonlinear regression. Literatures [18] expounded a chaos genetic algorithm based on chaos variable in genetic operation. It made the individuals of subgeneration distribute uniformly in the defined space and avoided its prematurity. Literatures [19] brought forward a eugenic evolution programming based on the feature analyses of deterministic algorithms and evolution algorithms. It integrates two-point gradients method and evolution programming in the evolutionary process of individuals. Furthermore the evolution operations are adjusted for global searching. All of these algorithms were successfully applied in kinetic parameters estimation in supercritical water oxidation process.

In chemical reaction, the removing rate is the most important index mark. Estimating the parameters exactly can figure out each factor's influence and consequently establish foundation of designing and optimizing industrial devices. 2-chlorine carbolic acid is a representational organic wastewater, and it can be decomposed to carbon dioxide, hydrochloride and water. The speed of the reaction depends on temperature, 2-chlorine carbolic acid concentration and oxidant concentration.

The objectives of our global kinetics analysis were to determine the Arrhenius parameters (A and E_a) and the reaction orders (a , b , and c) for 2CP, the rate

expression for 2CP disappearance during SCWO given in (14). These rate laws typically capture the general trends in the data, but they cannot be expected to capture the details of the complex oxidation chemistry.

$$Rate = A \exp\left(-\frac{E_a}{RT}\right) [2CP]^a [O_2]^b [H_2O]^c \quad (14)$$

Combining the rate law of (14) with the definition of conversion and the design equation for a constant-volume, plug-flow reactor leads to:

$$\frac{dX}{d\tau} = A \exp\left(-\frac{E_a}{RT}\right) [2CP]_0^{a-1} (1-X)^a [O_2]^b [H_2O]^c \quad (15)$$

The object function constitutes sum of squares of deviation between measured value and calculated value. By means of integrate (15) with initial condition $\tau=0$, $X=0$, the transformed equality is (16). The experimental data reference to literature [17].

$$\begin{aligned} & [(1-X)^{1-a} - 1] = \\ & (a-1)A \exp\left(-\frac{E_a}{RT}\right) [2CP]_0^{a-1} [O_2]_0^b [H_2O]_0^c \tau \end{aligned} \quad (16)$$

Suppose sample size is n , the removing rate of 2-chlorine carbolic acid in i^{th} sample is X_i . The sum of squares of deviation record as:

$$MSE(a, b, c, A, E_a) = \sum_{i=1}^n (X_i - X'_i)^2 \quad (17)$$

Parameters estimation is searching for A , E_a , a , b , and c which make (17), reaches the minimum. Optimization parameters in various literatures are listed in Table 3. In 20 runs, the IDE algorithm proposed in this article converges on required value every time.

Parameter	Literature [17]	Literature [18]	Literature [19]	IDE
a	0.88	0.8181	0.8067	0.8081
b	0.41	0.4750	0.4425	0.4444
c	0.34	0.3276	0.3336	0.3239
A	100	70.4	71.06	63.5416
E_0	46200	45153.2	46502	45625
MSE	0.2494	0.2225	0.2177	0.21769

Table 3: Optimization results comparison about reaction dynamics parameter.

From the Table 3, IDE algorithm is better than literatures [17]-[18] and corresponds to literature [19]. It's worthy to mention that convergence rate is very fast. In this experiment, the mean generation of get the prospective goal is forty in 30 runs. The results illustrate that IDE algorithm is just as effective in optimization of complex function.

6. Conclusions

An improved DE algorithm based on Alopex operator and memory selection was proposed. As can be seen from the previous results of the performance-testing experiment, with regard to the precision of global optimum and the convergence rate, IDE was clearly and consistently superior compared to original DE for global optimization of continuous space functions. Finally IDE was applied to reaction kinetic parameters estimation and also demonstrated its preferable capability.

Acknowledgment

This work is supported by National Natural Science Foundation of China for Distinguished Young Scholars (Grand No. 60625302), and by Shanghai Natural Science Foundation (Grand No. 06ZR14027).

References

- [1] R. Storn and K.V. Price, Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341-359, 1997.
- [2] J. Ilonen, J.K. Kamarainen and J. Lampine, Differential evolution training algorithm for feed-forward neural networks. In *Neural Processing Letters*, 7(2):93-105, 2003.
- [3] R. Storn, Differential evolution design of an IIRfilter. In *Proceedings of IEEE Int. Conference on Evolutionary Computation (ICEC96)*, New York, pp. 268-273, 1996.
- [4] P.S. Sastry, M. Magesh and K.P. Unnikrishnan, Two timescale analysis of Alopex algorithm for optimization. *Journal of Neural Comput*, 14(11):2729-2750, 2002.
- [5] L.N. De Castro and F.J. Von Zuben, Learning and optimization using the clonal selection principle. *IEEE Transaction on Evolutionary Computation*, Special Issue on Artificial Immune Systems, 6(3):239-251, 2002.
- [6] K.V. Price, R. Storn and J. Lampinen, *Differential evolution: A practical approach to global optimization*. Berlin: Springer-verlag, 2005.
- [7] J. Lampinen, A constraint handling approach for the differential evolution algorithm. In *Proceeding Congress on Evolutionary Computation (CEC 2002)*, Honolulu, Hawaii, pp. 1468-1473, 2002.
- [8] K. Price, Differential evolution vs. the functions of the 2nd ICEO. In *Proceeding of 1997 IEEE International Conference on Evolutionary Computation (ICEC 97)*, Indianapolis, USA, pp. 153-157, 1997.
- [9] D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos and M.N. Vrahatis, Parallel differential evolution. *IEEE Transaction*, pp. 2023-2029, 2004.
- [10] J.Y. Sun and Q.f. Zhang, DE/EDA: A new evolutionary algorithm for global optimization. *Journal of Information Sciences*, 169:249-262, 2005.
- [11] A.K. Qin and P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization. *IEEE Transaction*, pp. 1785-1791, 2005.
- [12] C.N. Patel, *Differential evolution-A method of global optimization*. Master dissertation, Department of Mechanical Engineering, Texas University, Arlington, 2002.
- [13] A. Bia, Alopex-B:A new, simpler, but yet faster version of the Alopex training algorithm. *Journal of Neural Systems*, 11:497-507, 2001.
- [14] A. Ratnaweera, S.K. Halgamuge and H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transaction Evolution Comput*, 8(3):240-255, 2004.
- [15] J.G. Digalakis and K.G. Margaritis, An experimental study of benchmarking functions for genetic algorithms. *Journal of Computer Math*, 79(4):403-416, 2002.
- [16] J. Liu and J. Lampinen, A fuzzy adaptive differential evolution algorithm. *2002 IEEE Region 10 Technical Conference on Computers Communications, Control and Power Engineering*, I-III:606-611, 2002.
- [17] R.k. Li, P.E. Savage and D. Szmukler, 2-Chlorophenol oxidation in supercritical water:global kinetics and reaction products. *Journal of AIChE*, 39(1):178-187, 1993.
- [18] X.F. Yan, D.Z. Chen, S.X. Hu and J.W. Ding, Estimation of kinetic parameters using chaos genetic algorithms. *Journal of Chemical industry and Engineering*, 53(8):810-814, 2002.
- [19] B. Zhang, D.Z. Chen and J. Rao, Estimation of kinetic parameters by using eugenic evolution programming. *Journal of Chemical Engineering of Chinese Universities*, 18(5):638-642, 2004.