Research Journal of Applied Sciences, Engineering and Technology 4(19): 3687-3691, 2012
ISSN: 2040-7467

Submitted: February 24, 2012 Accepted: April 08, 2012 Published: October 01, 2012

A Memetic Differential Evolution Algorithm with Adaptive Mutation Operator

[1,2]Fuqing Zhao, [1]Mingming Huo and [1]Weijun Ma
[1]School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China
[2]Key Laboratory of Contemporary Design and Integrated Manufacturing Technology, Ministry of Education, Northwestern Polytechnical University, 710072, China

Abstract: Differential evolution algorithm has been widely used, because of its efficient optimization and no complex operation and coding mechanism. But DE falls into the local optimum easily. So this study puts forward a memetic algorithm. The algorithm can increase the diversity of population and jump out the local extreme value point effectively. The convergence speed of the algorithm is improved, because of the self-adaptive operator which can adjust the scaling factor adaptively. Three classic benchmarks functions are used to evaluate the MMADE and basic differential evolution algorithm. The experimental results show that the MMADE algorithm is effective.

Keywords: DE, memetic algorithm, self-adaptive

INTRODUCTION

Optimization problems are widespread in the national defense, industrial, transportation, chemical industry, energy and communications and other aspects. The thought of these optimization problems is establishing the target function according to the problems. And then it gets the optimal solution and chooses the most reasonable scheme. In recent decades, many scholars optimized the traditional optimization algorithms. Such as: the method based on calculation, enumeration method, the simplex method and the Karmarc method which need to traverse the search space in dealing with large complex optimization problems and can't again the optimal solution in polynomial time. The scholars successively put forward many modern optimization algorithms, such as genetic algorithm, tabu search algorithm, PSO, the artificial neural network algorithm and Lagrangian relaxation method and other modern optimization algorithms. When differential evolution algorithm proposed, it got wide attention at once.

Differential Evolution algorithm (DE) was proposed by Storn and Price, (1997) in 1995 for solving Chebyshev polynomial. Differential evolution algorithm is a kind of heuristic random search algorithm. Because of its simple principle, robust, controlled parameters less and other advantages is concerned. Differential evolution algorithm not only effectively inherits the advantages of the genetic algorithm, but it has the different degrees improvement. Different with the genetic algorithm, the differential evolution algorithm utilizes real number coding. And its encoding and decoding mechanism are more simple and understandable. But it is similar with other commonly used evolutionary algorithms. Differential evolution algorithm also has the problems. Such as premature convergence and getting into the local superior easily (Holland, 1975; Yang *et al*., 2002).

At present the studies of differential evolution algorithm swing. Lianghong (Wu *et al*., 2006) put forward the Differential Evolution Algorithm with Adaptive Second Mutation. The algorithm added a new mutation operator to the best individual and part of the other individuals. And the variation operation is operated according to the size of the group fitness variance. It strengthened the ability of DE jumpping out of local optimal solution. Ji-Pyng and Feng-Sheng, (1998) added the accelerating operators and migratory operator in the standard differential evolution algorithm. The algorithm improved the diversity and convergence rate of the population. Ge *et al*., (2011) put forward the algorithm--An Adaptive Differential Evolution Algorithm Based on a Multi-population Parallel. To achieve the local search ability and the global search ability balance, the algorithm divided the individuals into different offspring population according to the individual fitness. At the same time it introduced the mutation operator corresponding population. So the problem was changed into multi group parallel problems.

Memetic Algorithm (MA) was proposed by Pablo Moscto (Moscato, 1989), which was inspired by Darwin's theory of evolution by natural and dawkins' cultural evolution Algorithm which is suitable for the local search and in the simulation of cultural evolution Algorithm based on the optimization Algorithm. Meme is the method

Corresponding Author: Fuqing Zhao, School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China

of improving the individual of the population, such as the disturbance, local search, etc., MA is in essence a unit based on individual local heuristic search and population global search. In recent years the MA becomes a hotpot in the field of evolutionary computation.

In the evolutionary process, with the increase of the iteration, the diversity of the population decreased is the root causes for the precocious of DE easily. This study first introduces the standard difference evolution algorithm. In the light of the existing problems of DE, this paper puts forward a kind of Memetic algorithm with self-adaptive mutation operator. This paper compares and analyses the proposed algorithm and the standard differential evolution algorithm through the matlab simulation experiments.

THE DESCEIPTION OF THE ALGORITHM

Classic Differential Evolution(DE): The basic idea of DE: The algorithm operates the initialed individual of the population randomly on variation operation and produces the individual variation vector. Then the algorithm applies the cross operation to the variation vector and target vector and produces test vector. Through the "greedy" selection operation, thus it produces a new generation of the population. Optimization problems:

$$Min\ f(x_1, x_2, ..x_D)$$
$$s.t.\ x^L_j \le x_j \le x^U_j = 1, 2, ..., D$$

where, D is the solution space dimension. x^L_j , x^U_j respectively expresses the upper and lower bounds of the j^{th} part of the scope x_j . The process of DE algorithm as follows (The flow chart of DE is shown in Fig. 1):

- **Initial population:** The initial population:

$$P^0_{i,j} = rand(1) \times (x^U_{i,j} - x^L_{i,j}) + x^L_{i,j} \quad (1)$$

rand (1) produces random numbers among (0, 1).

- **Mutation operation:** The biggest characteristic that Differential evolution algorithm is different from the other algorithms is the mutation operation. It leads the difference direction by the differential vector. It controls the step length by the scaling factor.
The formula of mutation is:

$$v^{G+1}{}_i = x^G{}_{r1} + F(x_{r2}{}^G - x_{r3}{}^G) \quad (2)$$

Among them: $r_1\ r_2\ r_3 \in \{1,2, ..., Np\}$ and $r_1 \neq r_2 \neq r_3 \neq I$; $x^G{}_{r1}$ is called the based vector of the father generation; $x_{r2}{}^G - x_{r3}{}^G$ is called the differential vector of the father generation; F is the scaling factor; Np is the scale of the population; G is the generation of the current population; G+1 is the following generation.

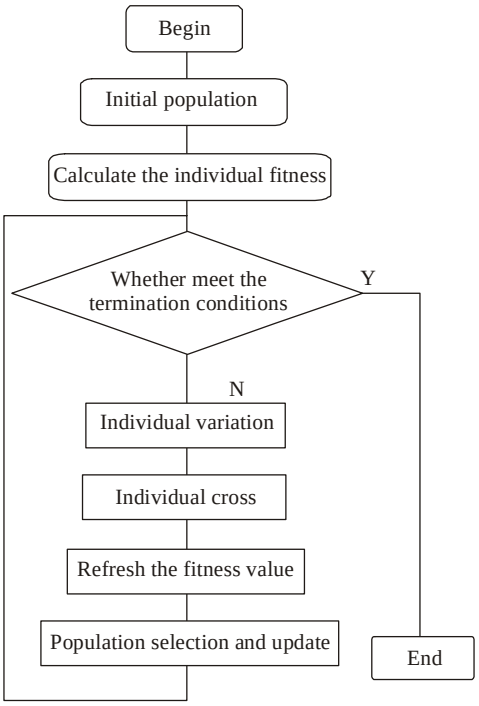

Fig. 1: The standard differential evolution algorithm flow chart

- **Crossover operation:** Using the variation vector $v_i{}^{G+1}$ and $x_i{}^{G+1}$ which produced by the formula (2) produces the test vector by the formula (3).

The formula of crossover operation is as follows:

$$u^{G+1}_{i,j} = \begin{cases} v^{G+1}_{i,j}, & if\ ((rand(j) \le CRor(j = rnbr(j)) \\ x^{G+1}_{i,j}, & otherwise \end{cases} \quad (3)$$

where, CR is the cross factor; *rand* (j) is the uniform distribution random number among (0, 1) which is produced by the j^{th} evaluation; *rnbr* (j) is an integer which is selected randomly from{1, 2, ..., D}; jϵ{1, 2, ..., D, D is the dimensions of the problem number. The value of CR decides the contribution of $v_t{}^{G+1}$ to $u_{t,j}{}^{G+1}$ .When = 1, $u^{G+1}{}_{i,j}$. is completely contributed by $v_t{}^{G+1}$. This moment is in favor of local search and speeding up the search speed, but it easily falls into the local optimal against global search. When CR = 0, $u_{t,j}{}^{G+1}$ is completely contributed by $x_t{}^{G+1}$. This moment is in favor of maintaining the diversity of the population and to the benefit of the global search, but the convergence speed is slower. So it follows that convergence rate and diversity is a pair of encompasses.

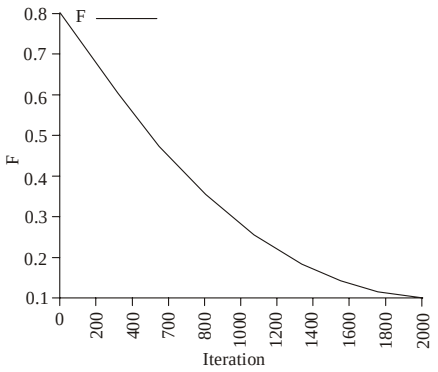

Fig. 2: The curve of scaling factor with the iteration

- **Selection operation:** The formula of selection strategy is as follows:

$$`x_i^{G+1} + \begin{cases} x_i^G, & if\ (f(u_i^{G+1}) \geq f(x_i^G)) \\ u_i^{G+1}, & otherwise \end{cases} \quad (4)$$

In the formula f is the objective function; and with the objective function values is small for the best. x_t^{G+1} the individual of next generation. The above operation is executed repeatedly until it meets the corresponding iteration times or achieving the corresponding precision.

Self-adaptive scaling factor: Generally speaking, a good algorithm is required with strong global search capability in the initial stages, so as to find much more global optimal solutions; and has strong local search ability in later stages, so as to improve the algorithm convergence speed and accuracy. Scaling factor F is closely related to the convergence speed. How to make the scaling factor can not only help to the development of the solution space but to the convergence of the population. Liqiang *et al*. (2008) studied the strategies of scaling factor value of the differential evolution algorithm. The simulation results show that F with the increase of the iteration gradually reduce is favor of improving the algorithm convergence rate. This paper puts forward the self-adaptive scaling factor according to the thought of this (the convergence curve of scaling factor shows in Fig. 2).
The formula is as follows:

$$F = F_0 \times 2^{(\frac{G-1}{G\max})^2} \quad (5)$$

where, F_0 is the initial value of the scaling factor and its initial value is 0.2; G is the iteration; G_{max} is the maximum iteration. The self-adaptive scaling factor in the start of the algorithm the mutation rate $F = 2\ F_0$, with larger mutation rate, so as to keep the diversity of the individual; with the progress of the algorithm the mutation rate gradually reduces. The algorithm mutation rate closes to F_0 the later stage, so as to avoid the optimal solution being destroyed.

Neighborhood search strategy: This paper adoptees local search which nearby the optimal solution x_{best} with the distribution of Cauchy in each iteration after getting a optimal solution x_{best}. The search times is k. Then it compares the selected optimal solution $x_{best}^{'}$ with x_{best}, if $x_{best}^{'}$ better than x_{best}, then the value of x_{best} replaces with value $x_{best}^{'}$ and $x_{best}^{'}$ replaces a random population individual. End the search. Otherwise continue DE basic operations. Because after each generation of evolution the algorithm adds local search operation to the population, so the algorithm convergence speeds is improved, but this may run into local optimal. So this paper introduces a fitness variance strategy. The concept of fitness variance:

Definition: The population size is Np . f_i is the individual fitness value of the i th individual. $\bar{f}$ is the:

$$\sigma^2 = \sum_{i=1}^{N_p} ((f_i - \overline{f}) / f) \quad (6)$$

where, f is the normalized factor. Its value is:

$$f = \begin{cases} \max\{|f_i - \overline{f}|\}, & \max\{|f_i - \overline{f}|\} \geq 1 \\ 1, & otherwise \end{cases} \quad (7)$$

whether the group fitness variance σ^2 is smaller than a certain threshold w to judge whether happened premature convergence. The value of w in literature has not a fixed value standard, but through the research found that premature convergence existed certain variance correlation with group fitness: When there is premature convergence, the value iteration little change in continuous several times. So if group fitness variance in the adjacent iteration changes little, it may appear premature convergence. At this moment it need to add the disturbance in the current population to jump into local minima. The neighborhood search added in this paper is:

$$x_{best}^{'} = x_{best} + 0.618C \quad (8)$$

where, x_{best} , $x_{best}^{'}$, respectively represent the optimal value of current generation and the optimal value of times search every time; C obeys the Cauchy distribution which mean value is zero. When fitness

variance is less than the threshold or the fitness variance adjacent small differences in search, it takes *k* times local search around the optimal value of the population.
The realizing procedure of the Memetic DE With self-adaptive scaling factor is as follows:

Step 1: Initial population scale *Np*; Minimum scaling factor F_{min}; Maximum scaling factor F_{max}; Crossover probability factor CR; Maximum iterations G_{max}; Upper and lower bounds of the population H_j, j $L_{i,j}$, $i \in \{1, 2, ..., Np\}$; Local search times k .

Step 2: Initialize the population according to the formula

(1): $P_{i,j}^0 = rand(1) \times (x_{i,j}^U - x_{i,j}^L) + x_{i,j}^0$

rand (1) produces random Numbers among (0, 1).

Step 3: Evaluate the initial population and find out the best individual fitness value recorded as x_{best} ,and record the fitness value f_{best}, set evolution algebra 1, that is $G = 1$.

Step 4: Use the formula (5) to execute the mutation, get the individual variation $v^{G+1}{}_i$.

Step 5: Use the formula (3) to execute the crossover operation, get the test individuals $u^{G+1}{}_i$.

Step 6: Use the formula (4) to execute the select operation, get the individual $x^{G+1}{}_i$ of next generation.

Step 7: Execute the comparison operations. If $f(x_t^{G+1}) < f_{best}$, then $x_{best} = x_t^{G+1}$, $f_{best} = f(x_t^{G+1})$, turn to step 8. Otherwise continue.

Step 8: Execute Step 4 ~ Step7 *Np* times.

Step 9: Take up *k* times local search using the formula (8) and get x_k by calculating.

Step 10: Find out the individual of the best fitness value $x_{best}^{'}$ among x_k . If $f(x_{best}^{'}) < f_{best}$, then $x_{best} = x_{best}$ f_{best} , $= f(x_{best}^{'})$ and produce a random integer t $\in [1, Np]$, index(x_{best}) = t, $x_t^{G+1} =$, $x_{best}^{'}$ turn to step 11; Otherwise continue.

Step 11: Evolution algebra add 1, that is $G = G + 1$; if $G < G_{max}$, then turn to step 4, otherwise turn to step 12.

Step 12: Output results x_{best} and f_{best} .Numerical experiments:

NUMERICAL EXPERIMENTS

In order to test the performance of the algorithm in this paper more effectively, this paper chooses three typical Benchmarks functions and compares with DE. This paper chooses one of the typical Benchmarks functions: Sphere function, Griewank function, Schwefel function. The test functions related description Table 1.

Table 1: The list of parameters

Parameter names	Expressions	Parameters
Crossover probability factor	*CR*	0.1
Initial scaling factor	F_0	0.2
Local search times	*K*	20
Population scale	N_p	100
Maximum iteration times	G_{max}	200
Optimization times	*t*	20

Table 2: Typical benchmarks functions

Function names	Test function expressions	Value range	Dimension	Peak type
Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	[-100,100]	30	mono peak
Griewank	$f_2(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}} + 1$	[-600,600]	30	multi-peak
Schwefel	$f_3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	[-100,100]	30	multi-peak

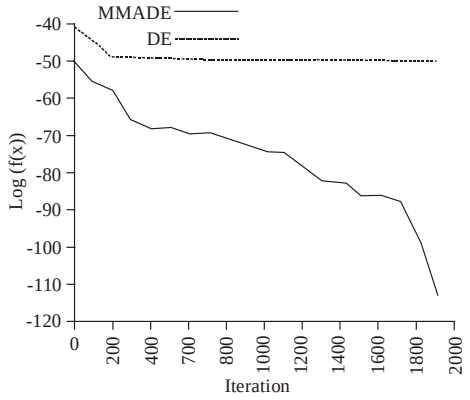

Fig. 3: The optimum curve of f_1

Benchmarks functions include the single-peak function and multi-modal function. Choosing above test functions and solving the functions' minimum value in the domain of definition can test the performance
of the algorithm from different aspects. The operating environment of the algorithm is: Dell Vostro 1088 matlab R2010b. The expressions of the test functions are as Table 1.

Experimental parameters are set as follows: Except that the maximum iteration times f_2 are 1000 *Np*, = 60, other functions of the population scale and the maximum iteration as Table 2. The experimental results are shown in Table 2. The contrast optimizes results of the simulation experiment are shown in Fig. 3 to 5.

In order to compare the convergence speed of MMADE and DE, the optimal values which are found by the algorithms MMADE and DE through 20 times are set by descending order. And optimal results are taken logarithms every time. The optimal curves are shown in Fig. 3 to 5. In these pictures, the y-coordinate is the

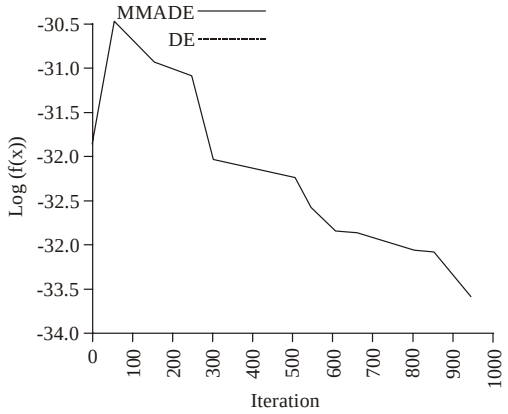

Fig. 4: The optimum curve of f_2

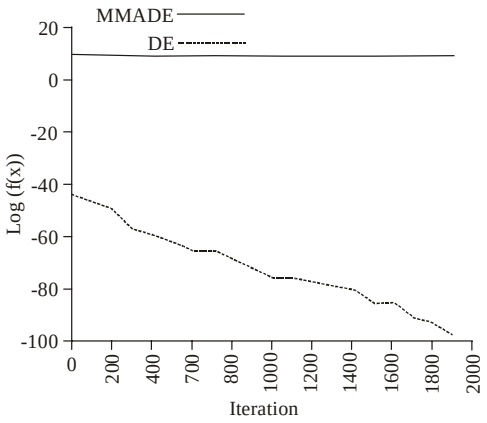

Fig. 5: The optimum curve of f_3

optimal value found by the logarithm. The horizontal ordinate is the evolutionary iteration. Blue solid line and red solid line respectively represent the optimal curves of MMADE and DE.

In the charts some of the curves disappear when they don't reach the maximum iteration. This states that in a moment the algorithm has found the global optimal value. Such as the optimal curves of f_2, f_2 do not appear the red lines, that is the algorithm finds the global optimal value after the first generation. From the general convergence speed, it can see that the convergence speed of MMADE is faster DE's.

From the analysis of above simulation results, it can see that the capability of searching and the convergence speed of MMADE are better than DE's.

CONCLUSION

This study introduces a self-adaptive Memetic algorithm based on the standard differential evolution algorithm. This algorithm uses the local search strategy of the Memetic. It increases the diversity of the population. It makes the population jump out local optimum and avoids the precocious phenomena effectively. The experimental results show that the algorithm puts forward in this paper MMADE relative to DE algorithm has higher precision and faster convergence speed.

ACKNOWLEDGMENT

This work is financially supported by the National Natural Science Foundation of China under Grant No.61064011. And it was also supported by Scientific research funds in Gansu Universities, Science Foundation for the Excellent Youth Scholars of Lanzhou University of Technology, Educational Commission of Gansu Province of China, Natural Science Foundation of Gansu Province and Returned Overseas Scholars Fund under Grant No. 1114ZTC139, 1014ZCX017, 1014ZTC090, 1114ZSB091 and 1014ZSB115, respectively.

REFERENCES

Ge, Y.F., W.J. Jin, L.Q. Gao and D. Feng, 2011. An adaptive differential evolution algorithm based on a multi-population parallel [J]. J. Natural Sci. Northeastern University, 32(4): 481-484.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems [M]. Michigan Press, Ann Arbor, pp: 1-50.

Ji-Pyng, C. and W. Feng-Sheng, 1998. A hybrid method of differential evolution with application to optimal control problems of a bioprocess system [A]. Proceeding of IEEE Evolution Computation Conference [C], pp: 627-632.

Liqiang, J., Q. Yingfeng and L. Guamgbin, 2008. A modified differential evolution algorithm for linear systm approximation. Electron Optics Control, 15(5): 35-37.

Moscato, P., 1989. On evolution, search, optimization, genetic algorithms and martial arts: toward memetic algorithms. Tech. Rep. Caltech Concurrent Computation Program, Rep. 826, Pasadena, California Inst. Technol, CA.

Storn, R. and K. Price, 1997. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces [J]. J. G local Optim., 11(4): 341-359.

Wu, L.H., Y.N. Wang and X.F. Yuan, 2006. Differential evolution algorithm with adaptive second mutation [J]. Contr. Decision, 21(8): 898-902.

Yang, Q.W., J.P. Jiang and Z.X. Qu, 2002. Improving genetic algorithms by using logic operation [J]. Contr. Decision, 15(4): 520-512.