IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 8 Issue 2, February 2021
ISSN (Online) 2348 – 7968 | Impact Factor (2020) – 6.72
www.ijiset.com

Solving Function Optimization Problem with the Differential Evolutionary Algorithm

Ziqiao Yu[1], Xiaoxia Zhang[2]
[1] College of Software Engineering, University of Science and Technology LiaoNing, Anshan 114051, China, patejuliacaqdj70@gmail.com

[2] Professor of College of Software Engineering, University of Science and Technology LiaoNing, Anshan 114051, China, zhangxiaoxia@ustl.edu.cn

Abstract
Differential Evolutionary algorithm (DE) is an evolutionary algorithm which is simply and strong effectiveness. And the DE algorithm is often used to solve the function optimization problem. Therefore, we selected two functions which are Apline and Schwefel Problem 22 to test the DE algorithm in this paper. Moreover, we also provide the illustrating procedure describe of DE algorithm. Finally, the performance of the DE algorithm can be proved from the results.
Keywords: *Differential Evolutionary Algorithm, Function Optimization Problem, Apline, Schwefel Problem 22.*

1. Introduction

There are a lot of algorithms used to work on the high dimensional function optimization problems [1],[2]. Now many nature-inspired meta-heuristic algorithms have been displayed such as Genetics algorithm (GA) [3], Bat Algorithm (BA) [4], Firefly Algorithm (FA) [5], Butterfly Optimization Algorithm (BOA) [6], Grey Wolf Optimization (GWO) algorithm [7], Particle Swarm Optimization (PSO) algorithm [8] and Differential Evolution (DE) algorithm [9].
In the past ten years, meta-heuristic algorithm has been widely used in solving the function optimization problems, especially in the face of high dimensions optimization problems. A number of evolutionary algorithms have gradually come to people's attention such as the DE algorithm, GA algorithm and so on. Among them, DE algorithm is one of the most powerful and interesting evolutionary algorithms, which can be effectively applied to the function optimization problem. The DE algorithm is like to the GA algorithm which allows each generation of solutions to evolve from previous generations. However, the main difference between the DE and GA algorithms, is the selection process and the unique variation process. All solutions have the same chance to be selected as independent fitness values in the DE. The selection process criterion adopted by the DE is the retention of competitive selection comparison between the new solution and the parent solution. This selection strategy provides a great advantage for the algorithm, and the performance is better than GA performance. The applications of the DE algorithm can be found in many studies. The traditional DE algorithm includes generating initial solution, mutation process (mutation process), crossover process (crossover process) and selection process (selection process). However, the convergence of traditional DE strongly depends on the selection of three main control parameters. If the difference strategy and parameter selection are improper, the algorithm will fall into the dilemma of local search or slow convergence speed.

2. Function Optimization Problem Description

Apline:

$$f(x)=\sum_{i=1}^{d}\left|x_i\sin(x_i)+0.1\,x_i\right| \qquad [-10,10] \tag{1}$$

Apline function which is a classic multi-modal minimization test. When it approaches infinity in the domain of definition, the function produces a large number of differentiable local extremum along the direction of the independent variable, which is difficult to optimize.

Schwefel Problem 22:

$$f(x) = \sum_{i=1}^{d} |x_i| + \prod_{i=1}^{d} |x_i| \qquad [-10,10] \tag{2}$$

Schwefel Problem 22 function has many local minimum to test the capacity of three algorithms which is jumping out of the local minimum and global searching. It can be proved whether the DE algorithm can jumps out of local minimum towards global minimum direction.

3. Differential evolutionary algorithm

Basic DE algorithm includes initial population, mutation process, crossover process and selection process which combine together form the evolutionary strategy of the DE algorithm. The setting of the whole evolutionary strategy takes the population as the operating object and determines the performance of the algorithm. Basic DE algorithm steps are as follows:

3.1 initial population

The first step is to initialize randomly within a certain range to produce a parent population. The population is a solution space composed of *NP* solution individuals, i.e. the space size is *NP*. Each solution individual consists of *D* individual variables, that is each solution individual is an array of $1 \times D$ dimension. *G* is the maximum number of iterations for the population. $X_i(g)$ expresses the *ith* individual in population *X* which is generated according to Equation (6).

$$X_i(g) = X_{min} + rand(0,1) * (X_{max} - X_{min}) \tag{3}$$

Where X_{min} and X_{max} represent the minimum and maximum limits of the value range of the $X_i(g)$. $rand(0,1)$ represents the random number between 0 and 1.

3.2 mutation process

In the mutation process, each individual from the current population is transformed into mutation individuals. Each mutation individual V_i(g) is generated by using Equation (6).

$$V_i(g) = X_{r1}(g) + F\left(X_{r2}(g) - X_{r3}(g)\right) \quad r1 \neq r2 \neq r3 \tag{4}$$

Where X_{Fg}(g) is the global optimization solution which has the best fitness value, X_{r1}(g) and X_{r2}(g) are two randomly selected individuals from the current population *X*(g), and *F* is the scaling factor which in between 0 and 2. The shrinkage level of the disturbing individual (X_{r1}(g) - X_{r2}(g)) depends on *F* and influences the searching direction for the global optimal solution in the DE algorithm.

3.3 crossover process

When the mutation process is finished, the crossover process starts. In the crossover process, the operating objects include the initial and mutation individual vectors of V_i(g)= [v_{i1}(g), v_{i2}(g), ..., v_{im}(g)] and X_i (g)= [x_{i1}(g), x_{i2}(g), ..., x_{in}(g)]. A binomial crossover scheme is adopted to generate a trial individual by using Equation (7).

$$u_{i,j}(g) = \begin{cases} v_{i,j}(g) & if \quad rand(0,1) \le CR \cup j = j_{rand} \\ x_{i,j}(g) & otherwise \end{cases} \tag{5}$$

Where p_c is a crossover factor which lies between 0 and 1. *rand* (0,1) is a random number between 0 and 1. If *rand* (0,1) is less than p_c, then the variable from the mutation individual is selected to form the trial individual. It can avoid the stagnation of search due to the population homogenization.

3.4 selection process

The DE algorithm uses one to one competition between parents and offspring to greedily select. Equation (8) describes the selection strategy which chooses the individuals of population for the next generation from the corresponding trial individuals in the current generation.

$$X_{i,}(g) = \begin{cases} U_{i,}(g) & if\ f(U_i(g)) < f(X_i(g)) \\ X_{i,}(g) & otherwise \end{cases} \quad (6)$$

The searching is stopped when the iteration g exceeds the maximum iterations G_m. Otherwise, the DE algorithm continues the searching for optimizing the population until meeting with all the conditions. The specific pseudo code of DE algorithm as follow in Fig 1.

```
Algorithm 1:DE algorithm
  Input: Population:P;Dimmension:D;Genetation:G
  Output: The best vector(solution)-Δ
  g<- 0(initialization);
 for i=1 to P do
    for j=1 to D do
       x_{i,j}(0) = x_{j,i}^L + rand(0, 1) * (x_{j,i}^U - x_{j,i}^L) ,
    end
end
Whlie (|f(Δ)|)or(t<T) do
    for i=1 to P do
       (Mutation and Crossover)
       for j=1 to D do
          v_{j,i}(g)=Mutation(x_{j,i}(g));
          u_{j,i}(g)=Crossover(x_{j,i}(g),v_{j,i}(g));
       end
       (Greedy select)
       If f(U_i(g)<f(X_i(g))) then
          X_i(g)<-U_i(g);
          If f(U_i(g)<f(Δ)) then
             Δ<-X_i(g)
          end
       else
          X_i(g)<-X_i(g);
       end
    end
    g<-g+1;
end
return the best vector Δ;
```

Fig. 1 DE algorithm Pseudo code

4. Computational Results

In this paper, the algorithm is tested by MATLAB R2017b. And we use Apline and Schwefel Problem 22 functions. The dimensions of them are 10 and 20. In table2 , there are six columns which show the function name, dim, *opt*, *best, average* and *Std* of two functions.

Table 2: DE simulation outcomes in two functions

Function name	*dim*	*opt*	*best*	*average*	*Std(%)*
Apline	10	0.00	7.00E-06	5.39E-04	7.54E-04
Apline	20	0.00	6.40E-02	3.70E-01	2.29E-01
Schwefel problem 22	10	0.00	1.53E-03	2.89E-02	1.95E-03
Schwefel problem 22	20	0.00	8.99E-02	1.39E-01	3.02E-02

From Table 2, it can be seen that the optimal values obtained by DE algorithm are closer to the classical optimal values whatever in 10 dim or 20 dim. Therefore, the single DE algorithm has good performance in Apline and Schwefel problem 22 functions.

5. Conclusions

In this paper, we proposed a specific algorithm illustration of original DE algorithm for solving the function optimization problem. By analyzing the results of two functions, DE algorithm has advantages of strongly global search capacity. Meanwhile, due to the strongly global search capacity, the DE algorithm will also easily trapped into the local optima. Therefore, it be proved that DE algorithm is further worth to studying and improving in function optimization problem.

Acknowledgments

This work it was supported by Project of Liaoning Xincheng Co., Ltd (Grant No. L20170989) and National College Students' innovation and entrepreneurship training program project (No. 202010146483)

References

[1] X.-S. Yang, A. H. Gandomi, S. Talatahari, and A. H. Alavi, Metaheuristics in Water, Geotechnical and Transport Engineering. Waltham, MA, USA: Newnes, 2012.
[2] A. H. Gandomi and A. H. Alavi, ‘‘Multi-stage genetic programming: A new strategy to nonlinear system modeling,’’Inf. Sci., vol. 181, no. 23, pp. 5227–5239, 2011.
[3] D. E. Goldberg and J. H. Holland, “Genetic algorithms and machine learning,’’Mach. Learn., vol. 3, nos. 2–3, pp. 95–99, 1988.
[4] X. S. Yang, ‘‘A new metaheuristic bat-inspired algorithm,’’ in Nature Inspired Cooperative Strategies for Optimization (NICSO).Springer, 2010, pp. 65–74.
[5] X-S. Yang, ‘‘Firefly algorithm, stochastic test functions and design opti-misation,’’ Int. J. Bio-Inspired Comput., vol. 2, no. 2, pp. 78–84, 2010.
[6] S. Arora and S. Singh, ‘‘Butterfly optimization algorithm: A novel approach for global optimization,” Soft Comput., vol. 23, no. 3, pp. 715–734, 2018.
[7] S. Mirjalili, S. M. Mirjalili, and A. Lewis, ‘‘Grey wolf optimizer,’’ Adv. Eng. Softw., vol. 69, pp. 46–61, Mar. 2014.
[8] R. Eberhart and J. Kennedy, ‘‘A new optimizer using particle swarm theory,’’ in Proc. 6th Int. Symp. Micro Mach. Hum. Sci. (MHS), Oct. 1995, pp. 39–43.
[9] R. Storn and K. Price, ‘‘Differential evolution–A simple and effificient heuristic for global optimization over continuous spaces,’’ J. Global Optim., vol. 11, no. 4, pp. 341–359, 1997.

Ziqiao Yu is currently reading in College of Software Engineering, University of Science and Technology LiaoNing, Anshan 114051, China. She is a postgraduate in the third year.

Xiaoxia Zhang is currently working as a Professor at the College of Software Engineering, University of Science and Technology LiaoNing, Anshan 114051, China.