An improved Stud Genetic Algorithm using the Opposition-based Strategy

Hongwei Xu[1, a]

[1]Wulanchabu Vocational College, Wulanchabu Inner Mongolia, China

[a]email: xuhongwei110@yeah.net

Keywords: Opposition-based Strategy; Stud Genetic Algorithm; Optimization Algorithm

Abstract. This paper proposed an improved Stud Genetic Algorithm using the Opposition-based strategy (SGAO) to improve the performance of the traditional SGA and accelerate its convergence speed. In SGAO, we use opposition-based approach to initialize the population and to perform mutation with the aim to improve the quality of solutions. In experiments, we use some benchmark functions to the show the performance of the proposed approach and compare it with other algorithms such as genetic algorithm, different evolutionary, particle swarm optimization and stud genetic algorithm. Results show that SGAO has faster convergence speed and higher solution precision.

Introduction

Genetic Algorithm (GA) is an optimization algorithm based on the natural biological evolution proposed. It has been widely used in optimization domain. To overcome the defect of the traditional genetic algorithm which has a week local search capability and easy to fall into minimum, the researchers have proposed various improved genetic algorithms to improve the optimization performance of the traditional genetic algorithm [1] [2]. Stud genetic algorithm is proposed in the literature [1], whose main improvement is that in each generation the best individual in the current population and the remaining individuals will take crossover operation. It is beneficial to preserve the good genetic information to be shared with other individuals. The SGA algorithm is simple in principle, and its optimization capabilities have been demonstrated [1] [2] [3], but the convergence speed and optimization capability remains to be further improved.

Opposition is a method of improving the performance of optimization algorithms [4] [5]. In real life, the concept of opposition is everywhere, such as male and female, and positive and negative electrons. If there is no opposition, it is difficult to describe entities around. For evolutionary algorithm, the quality of the initial solution will have an important influence on the optimization efficiency. High quality initial solution set can make the algorithm to find the optimal solution quickly, and save the search time, so as to improve the search efficiency of the algorithm. The initial solution set of low quality will increase the search time, and affect the operation efficiency of the algorithm. Although the initial solution set is directly related to the quality of the convergence rate and solution, but it does not catch enough attention. In the common stochastic optimization algorithms, such as genetic algorithm, differential evolution algorithm (DE) and the Particle Swarm Optimization algorithm (PSO), the initial solution is randomly generated. To improve the quality of the initial solution set, in literature [5], the opposition is introduced into evolutionary computation, and a population initialization method based on opposition is proposed, which replaces the traditional random generation method of population by the way of population opposition. We also consider the random point and its opposite point in the evolutionary process. It is more effective than simply using the random method. In the literature [6], the opposition is applied to differential evolution algorithm which proves its validity. To further analyze the effectiveness of the introduction of the opposition. The theoretical analysis and experimental verification of the combination of opposition and soft computing are carried out in the literature [7].

This paper proposed a kind of Stud Genetic Algorithm using the Opposition-based strategy (SGAO). The introduction of the opposition strategy can effectively improve the solution efficiency of SGA, and it provides a new optimization tool for solving optimization problems.

Stud Genetic Algorithm using the Opposition-based Strategy

Opposition-based Strategy

The opposition takes into account the current point and its opposite point at the same time from which to choose a good one. The opposing points defined are as follows [5]:

Definition 1: Suppose $P(x_1,x_2,\ldots x_N)$ is a point in the N-dimensional space, x_1, x_2, …, x_N, $x_i \in [a_i,b_i]$, i=1,2,…N, then the opposite point can be expressed as $P_o(x_{o1}, x_{o2}, \ldots, x_{oN})$, where $x_{oi} = a_i + b_i - x_i$.

For each point in the N-dimensional space, there is only one oppositional point [5] [6] [7]. The literature [6] [7] has demonstrated that the algorithms using a randomly selected opposing point are faster. Experimental results also show that the opposition search can be more effective than the random search. The effectiveness of opposition has been validated by the DE algorithms [6]. Aiming at improving the performance of the DE algorithm, the literature [6] also proposed a dynamic jump operation based on opposite individuals, which can be expressed as

$$x_{oi}=\max_i{}^g+\min_i{}^g-x_i, \tag{1}$$

where the $\max_i{}^g$ and $\min_i{}^g$ are the maximum and minimum of the *i*-th dimension of the *g*-th generation population respectively.

Stud Genetic Algorithm

Stud genetic algorithm is similar to the traditional genetic algorithm, the main difference is one of its parent in the crossover operation is fixed. That is, the best individual in the current population is called Stud. Then it will choose another parent from the remaining individual to perform crossover operation, such that good genetic information in the stud can be obtained quickly. It will help to improve the searching speed. The crossover operation can be used in two ways, i.e. one single-point crossover and two-point crossover. One-point crossover randomly generated a crossover operation as the starting position, and the ending position is fixed to the end of the chromosome. The two-point crossover is more flexible than one-point crossover, and its starting and ending positions of the crossover are randomly generated.

SGAO

The main idea of SGAO algorithm is as follows: To improve the quality of the initial solution set, it introduces opposition-based initialization in SGA population initialization phase. The algorithm randomly generates a prescribed numbers of candidate solutions. Each candidate solution's opposite solution is calculated according to Definition 1, then compares the fitness of the candidate solution with its opposite solution, finally puts the one with better adaptability in the initial solution set.

This paper proposed to use the opposition-based population initialization approach to initialize the population. The Pseudo-code is shown as Algorithm 1.

```
Algorithm 1: Population initialization in SGAO.
1. Randomly initialize each individual in the population $P=(P_1,P_2,\ldots,P_{Np})$
//$N_p$ is the population size. $P_i$ is the $i$-th individual in $P$
2. Calculate the fitness value of each individual $P_i$ ($i$ in the set {1,2,.., $N_p$ })
3. NFC = $N_p$
//NFC is the number of calls of the fitness functions
4. for $i$=0 to $N_p$ do
5.    for $j$=0 to $D$ do
6.       $P^*_{ij} = x^*_{ij} = a_j + b_j - x_{ij}$
7.       if $x^*_{ij}$ is out of the box-constraint $[a_j,b_j]$ then
8.          $x^*_{ij}=x_{rj}=a_j+rnd(0,1)(b_i-a_i)$
9.    end if
10.end for
11.Calculate the fitness value of $P^*_i$
```

```
12.NFC++
13. $P^* = P^* \cup P^*_i$
//$P^*$ is the opposite population of $P$ using the dynamic optimum and $P^*_i$ is the $i$-th individual in $P^*$
14.end for
15.Select $N_p$ fittest individual from $P \cup P^*$ as an initial population $P$
```

So the framework of SGAO is shown as Algorithm 2.

Algorithm 2: The SGAO algorithm.

```
1. Use algorithm 1 to generate the initial population
2. while NFC < MNFC do
//MNFC is the maximum number of calls of the fitness function
3. select the best individual $P_{best}$ from $P$
4. for $i$=0 to $N_p$ do
5.    if($P_{best}$!=$P_{it}$) {
6.        for $j$=0 to $D$ do
7.          if $rnd(0,1)<cr$ then
8.          //$cr \in (0,1)$ is the crossover probability
9.              $U_{ij} = V_{ij}$
10.         else
11.             $U_{ij} = P_{ij}$
12.         end if
13.         Calculate the fitness value of $U_i$
14.         NFC++
15.         if $f(U_i) \le f(P_i)$ then
16.             $P_i^{'} = U_i$
17.         else
18.             $P_i^{'} = P_i$
19.         end if
20.       end for
21.     }
22.     end if
23. $P = P^{'}$
24.   Use algorithm 2 to perform generation jumping operation
23. end for
24.end while
```

Experiment Study

In this section, we will use benchmark test functions to investigate the effectiveness of the proposed SGAO in the improvement of the final solutions. We will compare SGAO with DE, GA, PSO and SGA, and conducted controlled experiments. Our experiments were carried on a PC at 2.3GHz with 4GB of RAM.

In the following subsections, we provide details on the test functions of our study, parameter settings, and our experiment results and analysis.

Test Functions

In order to test the optimization ability of SGAO, this paper selected four typical test function included Ackley, Griewangk, Rastrigin and Sphere to test the performance of the algorithm.

✧ Function 1: Ackley Function

$$f_1(x) = -20 * e^{-0.2*\sqrt{\frac{1}{n}\sum_{j=1}^{n} x_j^2}} - e^{\frac{1}{n}\sum_{j=1}^{n}\cos(2\pi x_j)} + 22.71282, \ |x_j| \le 5$$

✧ Function 2: Griewangk Function

$$f_2(X) = \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{N} \cos(\frac{x_i}{\sqrt{i}}) + 1, |x_i| \le 600$$

✧ Function 3: Rastrigin Function

$$f_3(X) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2), \ |x_i| \le 5$$

✧ Function 4: Sphere Function

$$f_4(X) = \sum_{i=1}^{n} x_i^2, \ |x_i| \le 100$$

These four functions are very complex and have lots of local optimum which is hard to be solved by traditional approaches.

Parameter Settings

For comparison, the settings of the parameter are all set the same. The dimensions of all test functions are 20, the population size is 200, and the maximum evolution generation is 100. In order to overcome the randomness of the algorithm, each algorithm runs independently 50 times and the results are averaged. All algorithms are running on the same machine environment. In order to find more suitable way for cross-operation of SGAO, we used two different ways for crossover of SGAO to test the four test functions. And the jump probability j is 0.1.

Results and Analysis

According to the experimental data, for Ackley and Rastrigin test functions, when SGAO using two-point crossover, the convergence speed of algorithm is significantly faster than that using the one-point crossover. For Griewangk and Sphere functions, these two ways are similar, while the two-point crossover method is still better than the one-point. Therefore, in the following comparative studies, SGAO uses the two-point method for crossover.

In order to further test the performance of SGAO, we selected GA, DE, PSO and SGA for comparative study. And the parameters for GA, DE, PSO and SGA are set as the advice of literature [3]. Test results are shown in Figure 1 to Figure 4.

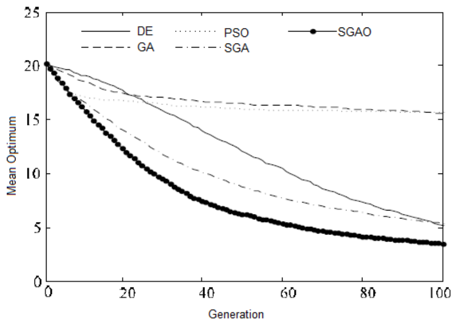

Figure 1 Evolutionary Curve of Ackley

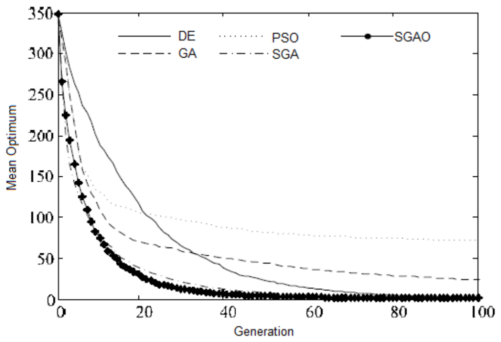

Figure 2 Evolutionary Curve of Griewangk

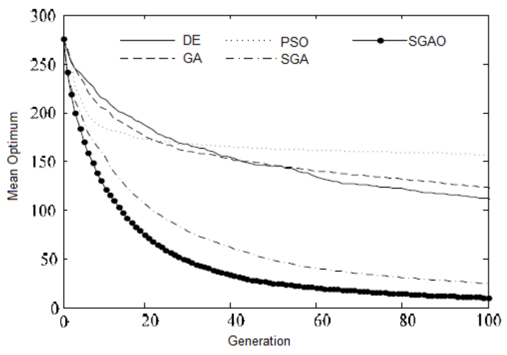

Figure 3 Evolutionary Curve of Rastrigin

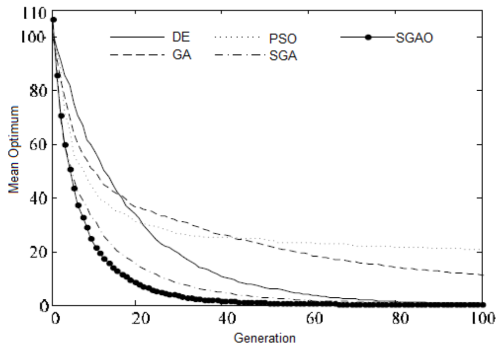

Figure 4 Evolutionary Curve of Sphere

As can be seen from Figure 1, for Ackley test function, GA and PSO algorithm will soon fall into local minima, and DE, SGA and SGAO can better avoid local minima. SGA convergences efficiency drops considerably as DE in the late evolutionary period, while SGAO remains good performance in the whole evolutionary stage. As it can be seen from Figure 2, although for Griewangk function, DE, SGA and SGAO have good optimum performance, but the convergence rate of SGA and SGAO was significantly faster than DE. As can be seen from Figure 3, for Rastrigin function, DE, GA and PSO are easy to converge to a local maximum, and SGA and SGAO showed a better optimization capability. From Figure 4 it can be seen that for the sphere function, in addition to the performance of PSO and GA is poor, some other algorithms also showed good optimization ability. But relatively speaking, SGAO still reflects its advantages. Its convergence speed is fast. The best solution and the average best solution of each algorithm which is run independently 50 times are shown in Table 1.

Table 1. Comparison of the final solutions. The results are the mean values over 50 runs.

F	DE	GA	PSO	SGA	SGAO
F1	5.08	15.51	15.66	5.32	3.47
F2	2.40	23.80	72.00	2.07	1.42
F3	111.19	122.26	122.26	24.41	9.78
F4	0.41	10.95	10.95	0.51	0.13

It can be seen in these types of algorithms, PSO shows the worst performance on the 4 test functions. The performance of DE and SGA is better, and its performance is not quite different against the other three functions in addition to the Rastrigin function. But the quality of SGA on Rastrigin function solution is far better than DE. In this paper, the SGAO algorithm shows good optimization ability on the 4 test functions, and the solution is the best solution or the average solution is better than the commonly used DE, GA and PSO algorithms. Compared with SGA, by introducing the opposition, optimization capability of SGAO has been further improved. For these test functions, the quality of the solution is significantly better than SGA.

Overall, for the above four test functions, the SGAO algorithm proposed in the current paper is obviously better than the other several typical optimization algorithms in convergence speed and solution precision. The reason is that the SGAO not only inherits the advantages of fast convergence of SGA, and due to the opposition introduced in the initialization phase of the population, it improves the quality of the initial solution set. The optimization efficiency of SGAO algorithm is further increased. While the introduction of dynamic opposition variation can increase the diversity of population, and reduce the possibility of their falling into local extreme values, so that the algorithm has a better optimization capability than usual optimization algorithm. In addition, this algorithm also inherits the advantages of SGA, i.e., the principle is simple and easy to implement.

Conclusion

In this paper, the opposition-based strategy is introduced into the genetic algorithm, and an improved genetic algorithm is proposed. The preliminary study on the test function shows that the introduction of the opposition-based strategy can effectively improve the optimization efficiency of SGA. Compared with other common optimization algorithms, SGAO has obvious advantages. We will further improve the optimization algorithm to improve efficiency and expand their specific application in engineering.

References

[1] Khatib W. and Fleming P. 1998. The stub GA: A mini revolution. The 5th International Conference on Parallel Problem Solving from Nature, New York, USA.

[2] Valceres V.R., Khatibb W., Fleming P.J. 2005. Performance Optimization of Gas Turbine Engine. Engineering Applications of Artificial Intelligence. 18(5): 575-583.

[3] Simon D. 2008. Biogeography-based optimization. IEEE Trasaction on Evolutionary Computation. 12(6): 702-713.

[4] Tizhoosh H.R. 2006. Opposition-based reinforcement learning. Journal of Advanced Computational Intelligence and Intelligent Informatics. 10(4): 578-585.

[5] Rahnamayan S., Tizhoosh H.R. and Salama M.M. 2007. A novel population initialization method for accelerating evolutionary algorithm. Computers and Mathematics with Applications. 53(10): 1605-1614.

[6] Rahnamayan S., Tizhoosh H.R. and Salama M.M. 2008 Opposition-based differential evolution. IEEE Transaction on Evolutionary Computation. 12(1): 64-79.

[7] Rahnamayan S., Tizhoosh H.R. and Salama M.M. 2008. Oppostion versus randomness in software computing techniques. Applied Software Computing. 8(2): 906-918.