Research on Function Optimization Based on Improved Genetic Particle Swarm Optimization

Hai Zhang, Shixin Li* and Xiaoyu Liu

School of Electronic Engineering, Tianjin University of technology and Education, Tianjin, China

*Corresponding author e-mail: 1459399409@qq.com

Abstract. Aiming at the shortcomings of single algorithm such as blind operation, directionless, long calculation time, low accuracy in Genetic Algorithm and poor diversity of population of Particle Swarm Optimization, which is prone to premature and fall into local optimum. In this study, the serial algorithm fusion idea is adopted, and the population evolved from the Genetic Algorithm is taken as the initial population to be executed by the Particle Swarm Algorithm, namely GAPSO algorithm. In view of the drawbacks of the respective algorithms in the execution of GAPSO algorithm, this paper proposes an Improved Genetic Particle Swarm Optimization algorithm, namely IGAPSO algorithm. In this algorithm, the improved GA only optimizes the initial particle once and then gives it to the improved PSO for optimization. Its optimization method and ability are greatly improved, resulting in a rapid increase in convergence speed. In addition, the improved PSO introduces adaptive inertia weight and learning factor, so that the particles can adaptively adjust global search and local search, and adaptively balance the influence of self-experience and social experience, which greatly avoids falling into local optimization problem and makes the solution more accurate. Through the verification of test functions, GAPSO has obviously improved the calculation accuracy, convergence speed and global stability compared with a single algorithm, while IGAPSO has also improved the convergence accuracy and speed compared with GAPSO, especially in multi-peak functions.

1. Introduction

Genetic Algorithm (GA) was proposed by Professor Holland of the United States in 1975 to solve optimization problems based on the natural selection phenomenon of survival of the fittest in biology. It is also called evolutionary algorithm. Its process is simple and is mostly used for function optimization, data mining, image processing, etc. However, there are some limitations, such as slow convergence speed and low convergence accuracy under multi-constraint conditions. Particle swarm optimization is an algorithm that optimizes the simulation of bird foraging process. The principle is simple, which can be realized by iteration of the velocity-displacement formula. In addition, there are few parameters to be adjusted and the particles have memory, which are mostly used for combination optimization, sensor network, vehicle scheduling, etc. However, with the continuous improvement of intelligent optimization algorithm, its shortcomings of easy falling into local optimization and difficult to deal with multi-dimensional problems are also exposed [1].
In recent years, many scholars have fused them. There are three main forms. First, parallel mixing, for example, Benvidi A [2] proposes GA-PSO parallel mixing to optimize ANN parameters. Secondly, embedded mixing, such as Sun Liping [3] embedding GA into PSO to improve the sample degradation problem of particle filter combined with Monte Carlo filtering and recursive Bayesian filtering, and the experiment proves the effectiveness; Third, serial mixing, such as Zhang J [4] proposed to stack PSO after GA algorithm is completed to realize clustering, although the effect is achieved, there are still problems such as slow convergence speed. This paper focuses on the research of GAPSO algorithm under serial fusion. In addition, an improved genetic particle swarm optimization algorithm is proposed.

2. Genetic Algorithm
GA uses population search technology. The objects it operates on and the solutions to the problems are all populations. A series of genetic operations are applied to the current population to generate a new generation of population, including selection of preferred strong individuals, crossing of new individuals by exchanging gene fragments between individuals, and mutation of new individuals by mutation of certain gene information. In this way, evolution is iterated step by step until the optimal solution is produced [5].

The steps of traditional GA are as follows:
(1) Initialization of variables
(2) Chromosome Sequencing
(3) Genetic manipulation and population merging and extraction
    ① The monarch scheme is adopted to carry out selective cross operation. ② Mutation operation based on probability. ③ Reorder of Chromosomes.
(4) Iteration

3. Particle Swarm Optimization Algorithm
PSO can be used in many optimization problems [6] by simulating the foraging process of birds to find the optimal solution. Each particle in PSO algorithm bears two values: position and velocity. The position of the particle represents a feasible solution, and the velocity is used to determine the direction and distance of the particle's motion. In addition, all particles update their positions and velocities by tracking their own empirical extremum and global extremum.

(1) Initialization of variables
(2) Initialization of population
(3) Update of population
    ① Updating the optimal position and optimal value of the individual. ② Update of global optimal position and optimal value. ③ Update of particle position and velocity. ④ Treatment of boundary conditions. Update the velocity and position of the particle with the following formula:

\[
v(j,:) = w \times v(j,:) + c1 \times \text{rand} \times (p(j,:) - x(j,:)) + c2 \times \text{rand} \times (g - x(j,:))
\]

\[
x(j,:) = x(j,:) + v(j,:)
\]

Where j represents a particle, j takes (1, N).
(4) Traversal

4. Genetic Particle Swarm Optimization and Its Improvement

4.1. GAPSO
The genetic particle swarm optimization algorithm in this study adopts serial fusion, that is, the execution result of GA is used as the initial matrix to execute PSO. In the initial stage, GA is used to optimize the randomly initialized particle swarm to improve the calculation accuracy, and then PSO is
used to uniformly move to the optimal solution to improve the convergence speed. This greatly avoids falling into local optimization, and can quickly and accurately find the optimal value in the individuals close to the optimal solution to realize global fast optimization [7].

4.2. **IGAPSO**

The improved genetic particle swarm optimization algorithm in this study is to optimize the position and velocity of randomly initialized particle swarm individuals once through improved GA and directly hand it over to PSO based on adaptive inertia weight and learning factor to improve the performance of genetic algorithm in global optimization and the shortcoming that particle swarm algorithm is easy to fall into local optimization. The specific operations are as follows:

4.2.1. **Initialization.** GA is used to optimize the initial position, velocity, optimal position and optimal value of each particle, and the optimized value is taken as the initial value for PSO.

First, the initial position, velocity, optimal position and optimal value matrix of a single particle are combined into A, and the overall ascending order is arranged into B by using sortrows function according to the initial optimal value (fitness value) of a single particle.

Then the first half of the particles with low fitness value are directly put into the next generation, and the second half of the particles are put into Crocs for a series of genetic operations:

First, it is a probability-based crossover operation. If the number randomly generated by the rand function is less than the crossover probability and one element in a row-by-column array generated by the randint function is 1, the second half of the particles are swapped in pairs with the element being 1 corresponding bit, thus realizing probability-based crossover.

Then there is the mutation operation based on probability. The product of the total number of particles and the mutation probability is taken as the iteration upper limit, the product of the particle dimension and the mutation probability is taken as the number of bits to be mutated, and then the randint selection of the particle volume element bits to be mutated from [1, N/2] and [1, D] is used to carry out inverse mutation, and the mutation operation based on probability is realized through repeated iteration.

4.2.2. **Optimization.** The replaced pbest value is compared with the initialized global optimal value gbest. If the pbest value is less than gbest value, the position corresponding to the pbest value is put into the global optimal position matrix g, temporarily being the global optimal position, and the gbest value at this time is also replaced by the gbest value.

4.2.3. **Update of Cross Section.** First, the positions that have undergone Cross mutation are substituted into the fitness function, and the function value is put into the optimal particle value in cross, and then compared with the corresponding values of the last N/2 particles in the original matrix B. If the function value is less than the corresponding value in B, the position corresponding to the value is put into the corresponding optimal position matrix in Cross, otherwise, the optimal position corresponding to the value of B is put into the corresponding optimal position matrix in Cross.

After the above update, the optimal position matrix of Cross is all the optimal positions of each particle, then these optimal positions are substituted into the fitness function, and the obtained values are put into the corresponding optimal values, thus obtaining the optimal position and optimal value of Cross.

Finally, Cross and the second half of b are merged, and sortrows is used to sort the optimal values in ascending order again. The first N/2 population obtained by sorting and the original particle population directly entering the next generation form the initial value for improving PSO execution.

4.2.4. **Docking with Improved Particle Swarm Optimization Algorithm.** The improvement of PSO in this study combines adaptive inertia weight and adaptive learning factor to improve the problem that particle swarm algorithm is easy to fall into local optimization.

(1) The introduction of adaptive inertia weight
Inertia weight $w$ balances the global exploration and local exploration of particle swarm optimization. According to the particle velocity update formula, the larger $w$ is, the larger the particle velocity is, so the algorithm will conduct global exploration with a larger step size, which is conducive to broadening the search area of the solution. The smaller $w$ is, the smaller the search step size of the corresponding particle is, and the particle performs fine search locally. In this study, an adaptive inertia weight $w$ is proposed. The specific improvement formula is as follows [8]:

$$W = \begin{cases} w_{\text{min}} - (w_{\text{max}} - w_{\text{min}})^k (f - f_{\text{min}}) / (f_{\text{avg}} - f_{\text{min}}), & f \leq f_{\text{avg}} \\ w_{\text{max}}, & f > f_{\text{avg}} \end{cases}$$

(3)

$w_{\text{max}}$ and $w_{\text{min}}$ respectively represent the maximum and minimum values of $w$. In this study, $w_{\text{max}} = 0.9$ and $w_{\text{min}} = 0.4$. $f$ represents the current fitness function value, $f_{\text{avg}}$ and $f_{\text{min}}$ represent the average fitness function value and the minimum fitness function value of all particles.

2. The introduction of adaptive learning factors

Through the speed update formula, we can know that the appropriate learning factor can effectively improve the efficiency of the algorithm and avoid falling into local optimization to the greatest extent. $c_1$ is a self-learning factor, which represents that the optimal position of an individual is affected by itself. $c_2$ is the social learning coefficient, which represents that the individual optimal solution is influenced by the global optimal solution. Moreover, at the beginning of searching for the global optimal solution, the algorithm is more inclined to search itself, so $c_1$ should be larger. $c_2$ should be relatively large at the later stage of the search, which is favorable for the particles to approach the global optimal solution. The specific improvement formula [9] is as follows:

$$c_n = c_{n,\text{ini}} + \frac{c_{n,\text{fin}} - c_{n,\text{ini}}}{T} \times t$$

(4)

In the formula, $n$ takes 1, 2, $c_{n,\text{ini}}$ as the initial value of the learning factor, $c_{n,\text{fin}}$ as the final iteration value of the learning factor, $t$ as the current iteration number, and $T$ as the maximum iteration number.

When the improved PSO interfaces with the above-mentioned improved GA, some operation methods, such as successively iteratively updating the individual's optimal position and value, updating the global optimal position and value, updating the individual's position and speed, processing the boundary conditions, and recording the global optimal values of past dynasties, are completely consistent with the above-mentioned traditional PSO.

5. Benchmark Function Simulation Test and Analysis

5.1. Benchmark function

The performance comparison of algorithms often needs benchmark functions to verify. In this study, two commonly used benchmark functions are selected: unimodal function Sphereh and multimodal function Rastrigrin [10].

1. Sphere function: $f(x) = \sum_{i=1}^{n} x_i^2$

2. Rastrigrin function: $f(x) = \sum_{i=1}^{n} (x_i^2 - 10 \cos(2\pi x_i) + 10)$
5.2. Simulation Test and Result Analysis

In this experiment, four algorithms of GA, PSO, GAPSO and IGAPSO are verified respectively by using benchmark function Sphere function and Rastrigrin function in MATLAB environment, in which the particle dimensions are 10, 20 and 30 respectively. The specific optimization process is shown in Fig. 1, Fig. 2 and Fig. 3. In addition, the average objective function value and average convergence iteration number of each algorithm on each function are plotted in the same table, such as Table 1 and Table 2, so as to compare their performance.

![Figure 1. Optimization process on Sphere function and Rastrigrin function when dimension is 10.](image)

![Figure 2. Optimization process on Sphere function and Rastrigrin function when dimension is 20.](image)

![Figure 3. Optimization process on Sphere function and Rastrigrin function when dimension is 30.](image)
Table 1. Average objective function value and average convergence iterations on Sphere function.

| Dimension | Algorithm | Average Objective Function Value | Average Convergence Iterations |
|-----------|-----------|----------------------------------|-------------------------------|
| 10        | GA        | 0.00040691                       | 121                           |
|           | PSO       | 3.7908e-025                      | 62                            |
|           | GAPSO     | 7.7676e-026                      | 44                            |
|           | IGAPSO    | 9.7632e-028                      | 39                            |
| 20        | GA        | 0.0051074                        | 161                           |
|           | PSO       | 1.5827e-010                      | 109                           |
|           | GAPSO     | 1.8972e-011                      | 71                            |
|           | IGAPSO    | 2.7292e-013                      | 70                            |
| 30        | GA        | 0.088974                         | 265                           |
|           | PSO       | 0.0060224                        | 202                           |
|           | GAPSO     | 0.0004202                        | 197                           |
|           | IGAPSO    | 0.0004121                        | 186                           |

Table 2. Average objective function value and average convergence iterations on Rastrigrin function.

| Dimension | Algorithm | Average Objective Function Value | Average Convergence Iterations |
|-----------|-----------|----------------------------------|-------------------------------|
| 10        | GA        | 6.9647                            | 258                           |
|           | PSO       | 12.9345                           | 263                           |
|           | GAPSO     | 0.080698                          | 242                           |
|           | IGAPSO    | 0.007346                          | 210                           |
| 20        | GA        | 34.832                            | 376                           |
|           | PSO       | 59.6974                           | 306                           |
|           | GAPSO     | 0.85464                           | 290                           |
|           | IGAPSO    | 0.43256                           | 276                           |
| 30        | GA        | 85.9664                           | 401                           |
|           | PSO       | 120.0869                          | 373                           |
|           | GAPSO     | 9.0572                            | 361                           |
|           | IGAPSO    | 9.0112                            | 342                           |

From the comparison and analysis of the above images and tables, it can be concluded that for single-peak functions, the optimization performance of each algorithm is good, and PSO is superior to GA in convergence accuracy and speed, while for multi-peak functions, the disadvantage that PSO is easy to fall into local optimization is exposed, and the convergence speed is fast, but the accuracy is far lower than GA. GAPSO combines the excellent performance of GA and PSO. Whether it is single-peak function or multi-peak function, it improves the convergence speed and accuracy compared with a single algorithm. The optimization performance of IGAPSO is further improved compared with that of GAPSO. In addition, it can be concluded that the convergence accuracy and speed on multimodal functions are not as good as unimodal functions. At the same time, the higher the particle dimension, the worse the optimization performance of each algorithm.

6. Conclusion

Combining GA and PSO in series can complement each other and benefit each other. For GAPSO, GA is initially used in the random initialization of particle swarm optimization to improve the calculation accuracy, and then PSO is used to uniformly move to the optimal solution to improve the convergence speed. However, the disadvantage of this algorithm is that their shortcomings still exist, which leads to premature convergence or low accuracy of the fusion algorithm occasionally. The improved genetic particle swarm optimization algorithm proposed in this study greatly improves the optimization performance. Due to the improvement of the optimization method and optimization ability of the
improved GA in the improved algorithm, the optimization times are greatly shortened and the convergence speed is greatly improved. In the improved PSO, adaptive inertia weight and learning factor are introduced, so that particles can adaptively adjust global search and local search, adaptively balance the influence of self-experience and social experience, greatly avoid the problem of the algorithm falling into local optimization, and make the solution more accurate.

**Acknowledgments**

This work was financially funded by Tianjin graduate research innovation project (2019YJSS194).

**References**

[1] Hongya Li, Yuzhong Peng, Chuyan Deng, et al. Review on Hybrid Research of GA and PSO [J]. Computer Engineering and Application, 2018, 54 (02): 20-28+39.

[2] Ali Benvidi, Saleheh Abbasi, Sajjad Gharaghani, Marzieh Dehghan Tezerjani, Saeed Masoum. Spectrophotometric determination of synthetic colorants using PSO-GA-ANN [J]. Food Chemistry, 2017, 220.

[3] Sun Liping. Research on Particle Filter Algorithm Based on GA-PSO [D]. Harbin University of science and technology, 2013.

[4] Zhang J. New clustering algorithm based on evolutionary computation [J]. Computer Engineering & Applications, 2011, 47 (24): 111-114.

[5] Ziyang Bao, Jizhou Yu, Shan Yang. Intelligent Optimization Algorithm and Its MATLAB Example [M]. 2nd Edition. Beijing: Electronics Industry Press, 2018.

[6] JAMES Kennedy, RUSSELL C, EBERHART. A Dircrete Binary Version of the Particle Swarm Algorithm [C]//IEEE International Conference on Systems, 1997: 4104-4108.

[7] Rebecca Moussa, Danielle Azar. A PSO-GA approach targeting fault-prone software modules [J]. The Journal of Systems & Software, 2017, 132.

[8] Guiqin Yu, Liudong Li, Yongfeng Yuan. A Hybrid Particle Swarm Optimization Algorithm Combining Adaptive Inertia Weight [J]. Journal of Harbin university of science and technology, 2016, 21 (03): 49-53.

[9] Jianxin Chen. Study on Function Optimization Based on Improved Particle Swarm Optimization [D]. Changchun University of science and technology, 2018.

[10] Yanglei Liu. Improved particle swarm algorithm based on quantum genetic algorithm and its application [D]. Harbin engineering university, 2011.