A Particle Swarm Optimization Algorithm Based on Dynamic Adaptive and Chaotic Search

Wei Zhuoa, Xuejun Yub
Faculty of Information Technology, Beijing University of Technology, Beijing, China
a18810819561@163.com, byuxuejun@bjut.edu.cn

Abstract. Although the particle swarm optimization algorithm has the advantages of fast convergence, easy to use and strong versatility, the algorithm also has the defects of low search precision, poor local search ability and easy to fall into local optimal solution. Therefore, this paper proposes a particle swarm optimization algorithm based on dynamic adaptive and chaotic search to ensure the global search ability of the particle swarm while avoiding falling into the local optimal solution. The experimental results show that compared with the comparison algorithm, the DACSPSO has stronger global search ability, higher convergence precision, and can effectively avoid premature convergence.

1. Introduction
Particle Swarm Optimization (PSO) is an emerging evolutionary optimization algorithm based on swarm intelligence [1, 2]. Compared with most evolutionary optimization algorithms, it simplifies complex operations, and has the characteristics of fast convergence, few adjustment parameters, easy implementation, and versatility. Based on these advantages, particle swarm optimization is well suited for use in scientific research or engineering practice. However, in practical applications, the PSO has problems such as lack of diversity of particles in the late stage, reduced search accuracy, poor local search ability and easy to fall into local optimal solution, which leads to premature maturity [3]. Aiming at this, this paper proposes a particle swarm optimization algorithm based on dynamic adaptive and chaotic search (DACSPSO), which combines the convergence speed of dynamic adaptive with the accuracy of chaotic search to ensure its global search ability, and avoid falling into local optimal solutions.

2. Improved particle swarm optimization

2.1. Basic particle swarm optimization
The basic particle swarm algorithm is described mathematically: there is a D-dimensional search space, and the sum of the number of particles in the space is N. The individual i in the particle swarm can be described by three formulas: (1) the velocity formula of the i-th particle is \( V_i = [v_{i,1}, v_{i,2}, ..., v_{i,D}] \); (2) the position formula of the i-th particle is \( X_i = [x_{i,1}, x_{i,2}, ..., x_{i,D}] \); (3) the i-th The best position the particle has explored so far is \( L_i = [l_{i,1,1}, l_{i,2,1}, ..., l_{i,D}] \). At the same time, the best position that the entire population has explored so far is \( L_g = [l_{g,1}, l_{g,2}, ..., l_{g,D}] \). The velocity and position of all particles will be updated by the following formula:
In the above formula, \( i = 1, 2, ..., M \), \( j = 1, 2, ..., D \). \( \omega \) is the inertia weight factor used to regulate the search ability of particles. \( r_1 \) and \( r_2 \) are the learning factors of the particles and are usually set to normal numbers. \( r_1 \) and \( r_2 \) are random numbers evenly distributed in the interval \([0, 1]\).

2.2. Dynamic adaptive adjustment strategy

The parameters involved in PSO are very important for the execution and optimization efficiency of the algorithm. Especially when dealing with complex problems, the unreasonable parameter setting will have a serious impact on the performance of the algorithm. In view of the value of the inertia weight factor, there are mainly random weights, linear decreasing strategies, nonlinear decreasing strategies, fuzzy rule strategies, etc [4], but these several strategic methods have their own defects. At the same time, unsuitable learning factors will affect the ability of particles to self-renew and learn from global optimal values, which will cause deviations in the direction of motion [5]. In addition, the balance point of the PSO is determined by \( L_q \) and \( L_g \) [6]. The search ability of the PSO is generated by the constant update of the balance point. However, when the quality of \( L_g \) is not good, the equilibrium point appears to be stagnant, resulting in premature convergence of the population.

Based on the above analysis, this study proposes a dynamic adaptive adjustment strategy for the parameters of the algorithm. The method combines the fitness of the particles with the degree of aggregation [7], and assigns values to the inertia weights through different methods, so that \( \omega \) can be adaptively adjusted dynamically. Then, the acceleration coefficient is dynamically adjusted by the adaptive value method of the inertia weight factor, so that the algorithm can obtain excellent global search ability under the premise of ensuring its convergence speed. Finally, according to the communication between the particles, the balance point of the population is adaptively adjusted to increase the execution efficiency of the algorithm.

1) Adaptive adjustment of inertia weight factor

The global optimal value of the particle is better than the current fitness value of all the particles. Let the current fitness value of the i-th particle in the population be \( s_i \), the variance of the fitness function be \( D \), and the population size be \( N \) then the average value of the population fitness value \( s_{avg} \) and the degree of particles aggregation \( \varepsilon \) can be expressed by the following formula:

\[
s_{avg} = \frac{1}{N} \sum_{i=1}^{N} s_i
gives the average value of the population fitness value \( s_{avg} \) and the degree of particles aggregation \( \varepsilon \).

\[
D = \sum_{i=1}^{N} [s_i - s_{avg}]^2
\]

\[
\varepsilon = \sin (\tan^{-1} D)
\]

The larger the value of the aggregation factor \( \varepsilon \) is, the higher the dispersion degree of the particles is. The smaller the value of \( \varepsilon \) is, the larger the aggregation degree of the particles is, and the population will appear to be trapped in the local optimal solution. The particles whose particle fitness value is better than the average fitness value \( s_{avg} \) are selected, and the average value \( s_{avg} \) of the fitness value is obtained. The specific process for assigning \( \omega \) is as follows:

First, if the current population's \( s_q \) is better than \( s_{avg} \), then this part of the particles is a better performing particle, which is close to the optimal solution of the problem. The speed of this part of the particle should be reduced, so the value of the inertia weight factor should be reduced.
\[ \omega_{i} = \omega_{\text{max}} - \frac{(\omega_{\text{max}} - \omega_{\text{min}})|s_{i} - s'_{\text{avg}}|}{|s_{g} - s'_{\text{avg}}|} \]  

Then, if the current population \( s_{i} \) is between \( s_{\text{avg}} \) and \( s_{g} \), let the speed of this part of the particle decrease nonlinearly with the number of population iterations, thus ensuring that the algorithm has better global search ability in the early stage. In the later stage, fine local search capabilities can be obtained.

\[ \omega_{i} = \omega_{\text{min}} + (\omega_{\text{max}} - \omega_{\text{min}}) \frac{1 + \cos \left( \frac{k \pi}{2} \right)}{2} \]  

Finally, if the current population's \( s_{i} \) and \( s_{\text{avg}} \) are poor, then the performance of this part of the particle is poor in the entire population. For this part of the particle, the current speed of the particle can be dynamically adjusted by the current degree of aggregation \( \varepsilon \).

\[ \omega_{i} = 1 - \frac{1}{p \exp(-q \varepsilon)} \]  

In the formula, the role of \( p \) is to control the upper limit of \( \omega \), and the role of \( q \) is to control the adjustment ability of the formula.

(2) Adaptive adjustment of acceleration factor

In this paper, the acceleration coefficient is adaptively adjusted by the value of the inertia weight factor, giving the better particles a larger \( c_{1} \) and smaller \( c_{2} \), so that they have excellent local search ability; on the contrary, giving poorer particles smaller \( c_{1} \) and larger \( c_{2} \), so that it has excellent global search capabilities.

The expressions of the acceleration coefficients \( c_{1i} \) and \( c_{2i} \) of the \( i \)-th particle are:

\[ c_{1i} = \frac{\omega_{i}}{\omega_{\text{max}}} \times (\omega_{1} + \sqrt{\omega_{i}} + 1) \]  

\[ c_{2i} = \frac{\omega_{\text{max}} - \omega_{i}}{\omega_{\text{max}}} \times (\omega_{1} + \sqrt{\omega_{i}} + 1) \]  

(3) Adaptive adjustment of the balance point

In the PSO, the role of the balance point is to provide the correct direction and region for the population. After sorting the optimal positions of the individuals in order from superior to poor, the individual optimal position weighted average \( L_{m} \) of the previous \( m \) particles is obtained, which is used to replace the \( L_{g} \) of the PSO. The formula for getting \( L_{m} \) is:

\[ L_{m} = \sum_{p=1}^{m} L_{p} \eta_{p} \]  

\[ \eta_{p} = 1 / \left( s_{p} \sum_{k=1}^{m} \frac{1}{s_{k}} \right) \]  

In the formula, \( \eta_{p} \) is the weighting coefficient of the algorithm, and \( s_{p} \) is the fitness value of the optimal position \( L_{p} \) of the individual particles. Then use the weighted average \( L_{d} \) of the sorted \( L_{q} \) and \( L_{q-1} \) to replace the particle's optimal position \( L_{q}, L_{q-1} \) as calculated:

\[ L_{d} = \frac{(L_{q-1} s_{p} + L_{q} s_{p-1})}{(s_{p} + s_{p-1})} \]
Since more optimal positions of the particles are acquired by $L_d$ and $L_m$, updating the optimal position of any one of the particles will update the balance point. Since the balance point update becomes more frequent, the search efficiency of the algorithm is significantly improved.

2.3. Local search strategy based on chaos optimization

By introducing the dynamic adaptive adjustment strategy into the PSO, the algorithm can also have better global search ability under the premise of ensuring convergence speed. However, due to the poor local search ability of PSO, it is easy to fall into the local optimal solution during the running of the algorithm. Chaotic optimization technology has the characteristics of randomness, regularity and ergodicity [8, 9], so it can be introduced into the algorithm to solve the problem that PSO is easy to fall into local extremum. The local search process based on chaos optimization proposed in this paper is as follows:

(1) In this paper, the Tent chaotic map [10] is used to generate chaotic sequences. However, since the Tent map is the same as the Logistic map, it has unstable periodic points in the iterative sequence of the mapping model, which will lead to the failure of the algorithm search. In order to prevent this from happening, this paper introduces a random perturbation for the Tent mapping. The improved Tent mapping formula is:

$$C_{t+1} = \begin{cases} 2(C_t + \text{rand}(0,1)/10) & 0 \leq C_t < 1/2 \\ 2(1 - (C_t + \text{rand}(0,1)/10)) & 1/2 \leq C_t \leq 1 \end{cases}$$

(14)

In the formula, \( \text{rand}(0,1) \) is a random number in the range [0, 1].

(2) Firstly, the initial decision variable $C_{t,j}$ of the optimization problem is mapped to the chaotic space with the value range [0, 1] through the formula (15), and the initial value $f_{t,j}$ of the chaotic sequence is generated. The relevant formula is as follows:

$$f_{t,j} = \frac{c_{t,j} - c_{\text{min},j}}{c_{\text{max},j} - c_{\text{min},j}}$$

(15)

In the formula, \( j \) is the dimension of the decision variable, \( j = 1, 2, ..., n \). \( c_{\text{max},j} \) and \( c_{\text{min},j} \) represent the upper and lower bounds of the initial decision variables in the \( j \)-th dimensional space respectively.

(3) Then use the Tent chaotic mapping equation (14) to generate a new chaotic sequence $f_{t+1,j}$ for the initial value $f_{t,j}$ of the chaotic sequence. Convert chaotic variables into optimization variables by linear mapping as follows:

$$y_t = p + (q - p) \times x_t$$

(16)

In the formula, the upper and lower limits of the range of the optimization variable \( y \) are represented by \( q \) and \( p \), respectively. In the process of population exploration optimization, the particles update the motion through iteration, and the optimization variables are traversed in the interval of the problem to be solved, and the corresponding chaotic variables are traversed in the interval [0, 1].

(4) When the algorithm appears premature, the obtained chaotic sequence $f_{t+1,j}$ is remapped into the solution space of the original optimization problem by the inverse mapping of the formula (17), and a new decision variable $C_{t+1,j}$ is generated to obtain a new particle population. To replace particles with poor fitness values. This approach increases the diversity of the population and enhances the ergodicity of the algorithm itself. The formula (17) is as follows:

$$C_{t+1,j} = c_{\text{min},j} + f_{t+1,j} \cdot (c_{\text{max},j} - c_{\text{min},j}), j = 1, 2, ..., n$$

(17)
In the formula, $f_{t+1,j}$ represents a chaotic variable, which is determined by the formula (14), and $C_{t+1,j}$ is the current positional variable of the particle.

(5) After transforming the traversal carrier of chaotic motion into an optimization variable, chaotic perturbation is introduced to update the optimal position $L_k$ of the population. Obtain the optimal solution in the current population, introduce a chaotic perturbation quantity according to formula (18), then map the carrier into a new decision variable according to formula (17), and calculate the fitness value according to variable $C_{t+1,j}$. When the termination condition of chaos optimization is reached, the new solution is taken as the result of the local search and returned.

$$G' = \tau G + (1 - \tau) \times K'$$

In the formula, $\tau$ is an adjustment parameter with a value range between $[0, 1]$. $K'$ is the optimal chaotic vector generated by mapping the current optimal solution vector $C' = (c'_{1}, c'_{2}, ..., c'_{t})$ over the interval $[0, 1]$ by the formula (14).

2.4. Improve the flow of particle swarm optimization

Based on the above research, the specific steps and flow chart of DACSPSO can be drawn as follows:

Step1: Perform initialization of the algorithm. Set the population size $N$, the algorithm allows the maximum number of iterations $T_{max}$, the upper limit $\omega_{max}$ and the lower limit $\omega_{min}$ of the inertia weight, the search space dimension $D$ and other parameters;

Step2: Chaotic initialization of the velocity and position of the particle through the Tent mapping;

Step 3: Through the dynamic adaptive adjustment strategy, and based on the fitness value $\bar{s}_i$ of the particle, adaptively adjust the inertia weight factor, acceleration coefficient and balance point of the algorithm through formulas (6) to (13). Then update the speed and position of the particles;

Step4: Calculate the fitness value of the particle, and update the individual extreme value $\bar{G}_i$ and the global extreme value $G$;

Step 5: Determine whether the population has premature convergence. If this happens, go to Step6, otherwise, go to Step3;

Step6: Select the best pre-S particles according to the fitness value, and introduce the chaotic perturbation strategy. The initial decision $C_{t,j}$ of the particle is mapped to the chaotic space by the formula (15), and the initial value $f_{t,j}$ of the chaotic sequence is generated. Then, through the formula (14), $f_{t,j}$ is mapped into a new chaotic sequence $f_{t+1,j}$, and it is inversely mapped to the solution space of the original optimization problem by the formula (17), and a new decision variable $C_{t+1,j}$ is generated. Calculate the fitness value of each feasible solution $C_{t+1,j}$ in the original solution space, and select the top S solutions $C^*_t$ with the best performance;

Step7: Find the S particles with the worst fitness value in the population and replace them with $C^*_t$; And update the global extreme value $L$ according to $C^*_t$;

Step 8: Determine whether the current algorithm has a termination condition, and if yes, turn to Step 9 to end the algorithm; otherwise, go to Step 3 to continue execution;

Step9: Stop the algorithm and output the global optimal solution.
3. **Experiment and result analysis**

In this paper, four classical test functions are selected to test the performance of the DACSPSO. The experimental comparison algorithms include DACSPSO presented in this paper, DNSPSO [11], OLPSO [12] and basic PSO. In the experiment, the size \( M \) of the particle population is 20, the dimension \( D \) of the particle variable is 30, and the maximum number of iterations \( N_{\text{max}} \) of the algorithm is 1500. The optimization performance of the algorithm is evaluated by calculating the mean and standard deviation of the optimal fitness values obtained from 100 searches. The test results are shown in Figure 2.
Figure 2. Convergence graph of test function

By analyzing Figure 2, it can be found that the overall performance of the DACSPSO is the best, followed by the DNSPSO and the OLPSO, and the worst is the PSO. Within a certain number of iterations, DACSPSO, DNSPSO and OLPSO can converge to the target precision. The global search ability of DACSPSO is stronger, the convergence speed and convergence precision are higher, and the diversity of population is better. The PSO still performs poorly when iterating to 1500 times. Moreover, the DACSPSO can jump out of the local optimal solution and has the ability to avoid falling into premature convergence. Therefore, the effectiveness of the improved algorithm in this paper is proved.

4. Conclusion
Aiming at the shortcomings of the basic particle swarm optimization algorithm, this paper proposes a particle swarm optimization algorithm based on dynamic adaptive and chaotic search (DACSPSO). First, let the parameters of the algorithm have the ability of dynamic adaptive adjustment, and then introduce chaos optimization technology into the local search. This can effectively control the speed and search accuracy of the particles, while avoiding the local optimal solution while ensuring the global search ability of the algorithm. Experiments show that DACSPSO performs best compared to the comparison algorithm.

References
[1] Kennedy J, Eberhart R C. Particle swarm optimization[C]. Proceedings of IEEE International Conference on Neural Networks. Piscataway: IEEE Press, 1995: 1492-1498.
[2] Z. H. Zhan, J. Zhang, Y. Li, Chung SH. Adaptive particle swarm optimization. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, 2009, 39 (6): 1362-1381.
[3] Liu Y, Wang G, Chen H, et al. An improved particle swarm optimization for feature selection [J]. Journal of Bionic Engineering, 2011, 8 (2): 191-200.
[4] Baskar S, Suganthan P N. A Novel Concurrent Particle Swarm Optimization [A]. Proceedings of the 2004 Congress on Evolutionary Computation [C]. Piscataway, NJ: IEEE Press, 2004: 792-796.

[5] Zhou Hong, Zhang Sheng, Liu Linlan et al. Automatic generation of path test data based on GA-PSO algorithm [J]. Journal of Computer Applications, 2010, 27 (4): 1366-1369.

[6] Huang H, Qin H, Hao Z, et al. Example-based learning particle swarm optimization for continuous optimization [J]. Information Sciences, 2010, 182(1): 125-138.

[7] Liu Dong, Feng Quanyuan. Research on PSO Algorithm Based on Aggregation Control [J]. Control and Decision, 2011, 26 (12): 1917-1920.

[8] Ikeguchi T, Hasegawa M, Kimura T, et al. Theory and applications of chaotic optimization methods [M]. Innovative Computing Methods and Their Applications to Engineering Problems. Springer Berlin Heidelberg, 2011: 131-161.

[9] Yang Shengpei, Li Zhongyang, Chen Zhongyang. Particle Swarm Optimization Algorithm Based on Chaotic Search and Population Crossing [J]. Computer Simulation, 2016, 33 (06): 218-222.

[10] Liu Bo, Wang Ling, Jin Yihui, et al. Improved particle swarm optimization combined with chaos [J]. Chaos, Solitons Fractals, 2005, 25 (5): 1261-1271.

[11] WANG H, SUN H, LI C H, et al. Diversity enhanced particle swarm optimization with neighborhood search [J]. Information Sciences, 2013, 223 (2): 119-135.

[12] Zhan Z H, Li Y, Shi Y H. Orthogonal learning particle swarm optimization [J]. IEEE Trans on Evol Comput, 2011, 15 (6): 832-847.