Optimization of CSO algorithm based on adaptive inertia weight coefficient

Dongnan Suo 1

1 Datang Northeast Electric Power Test & Research Institute, Changchun, Jilin, 130000, China

*Corresponding author’s e-mail: suodongnan@163.com

Abstract: The traditional CSO algorithm is easy to fall into local extremum in optimization. In this paper, a CSO algorithm based on weight coefficient is proposed. In the CSO algorithm, the inertia weight coefficient is introduced into the hen position formula, and the learning factor influenced by the rooster is added to the chick position formula. Finally, using the idea of heredity, individuals with excellent fitness value are selected for crossover and mutation with a certain probability. Through the simulation comparison of five typical test functions, the simulation results show that the improved CSO algorithm can avoid local optimization, strengthen the global extreme value search ability, and improve the convergence speed and accuracy range of the algorithm.

1. Introduction
In the field of Engineering Science, the solution of most problems can be covered in numerical or combinatorial optimization problems through some ideas or rules. Swarm intelligence algorithm (SIA)[1] is a simulation to realize the optimal search of targets according to the living habits, behavior patterns, physical laws and hidden rules of various group creatures in the natural world, that is, a random optimization algorithm based on group construction.

In 2014, Meng et al. [2] Constructed different search methods with hierarchy based on the foraging phenomenon of roosters, hens and chicks in the group, and proposed a chicken swarm optimization (CSO). Compared with other algorithms, this algorithm has the advantages of simple principle, easy implementation and strong global search ability.

CSO algorithm has been widely used in multi classifier systems, cluster analysis and industrial fields. However, with the further deepening of research, it is found that in CSO algorithm, the solution accuracy and convergence speed are still insufficient, and it is easy to transfer the global solution to the local optimal solution[3].

2. CSO Algorithm
The Chicken swarm algorithm is a new swarm intelligence optimization algorithm. It is a bionic optimization algorithm that simulates the search behavior of chicken swarm and the hierarchy between populations. Like other swarm intelligence algorithms, the whole chicken group can be divided into many subgroups, and the individuals of the whole chicken group are assigned roles.

The roles are divided into rooster, hens and chickens. The number of rooster determines the number of subgroups. Each subgroup has and only has one rooster, multiple hens and several chickens. According to the size of the population, the proportion factor of hens and the proportion factor of chickens are determined. To determine the number of hens and chicks in each subgroup. Individuals in
the same subgroup determine their learning and following rules according to the hierarchical relationship. Individuals in the same hierarchy have a competitive relationship, and there is competition among different subgroups.

The Chicken swarm optimization is a hierarchical system and group behavior in the simulated population. The chickens are divided into roosters, hens and chickens. The roosters are the best individuals in the group, actively seeking food; the hens are accompanied by the rooster; the chickens follow the chicken mothers to find food. In order to simplify the complicated foraging process, the following rules are formulated:

a. In the chicken flocks, there are multiple subgroups, each of which includes an optimal rooster, a group of hens, and chicks.

b. How to group the flocks and determine the identity of the chickens depends on the fitness of the chickens themselves. The chickens with the best fitness serve as the roosters, and each rooster will become the leader in the group. Chickens with the worst fitness are designated as chickens, and other chickens are used as hens. The hens randomly selected the living group, and the relationship between the hen and the chick was randomly matched.

c. The rank order, dominance relationship, and parent-child relationship in the 3 groups remain unchanged, and these states are redistributed and updated every G generation.

d. Chickens in the flock follow the companion cock to find food, the chicks look for food around the mother, and the dominant chicken has an advantage in competing for food.

Assume that N_{R}, N_{H}, N_{C}, and N_{M} represent the number of roosters, hens, chickens, and chicken mothers, respectively. Throughout the flock, all virtual chickens search for food by location in D-dimensional space.

Compared with roosters with poor fitness, roosters with better fitness are more likely to obtain food. This situation can be simulated by:

\[
\begin{align*}
    x_{i,j}(t+1) &= x_{i,j}(t) + (1 + \text{Randn}(0, \sigma^2)) \\
    \sigma^2 &= \begin{cases} 
        1 & \text{if } f_i < f_k \\
        \exp\left(\frac{f_i - f_k}{|f_i| + \varepsilon}\right) & \text{otherwise}
    \end{cases}
\end{align*}
\]  

(1)

Randn(0, \sigma^2) is a Gaussian distribution, \( \varepsilon \) is used to avoid segmentation errors, and \( \varepsilon \) is the smallest constant in the calculation, and \( k \) is the index of another rooster.

The more dominant hens have an advantage in competing foods. These phenomena can be expressed mathematically as follows:

\[
\begin{align*}
    x_{i,j}(t+1) &= x_{i,j}(t) + c_1 \times \text{rand} \times (x_{r_1,j}(t) - x_{i,j}(t)) \\
    &\quad + c_2 \times \text{rand} \times (x_{r_2,j} - x_{i,j}(t)) \\
    &= \frac{(f_i - f_j)}{\text{abs}(f_i + \varepsilon)}
\end{align*}
\]  

(2)

Rand is a uniform random number on [0, 1], \( r_1 \) represents the best individual male in the group of the \( i \)-then, and \( r_2 \) represents any individual randomly selected from the chicken, except for the chicken, and \( r_1 \neq r_2 \).

The chicken moves around the chicken mother to find food. The rules are as follows:

\[
\begin{align*}
    x_{i,j}(t+1) &= x_{i,j}(t) + F \times (x_{m,j}(t+1) - x_{i,j}(t))
\end{align*}
\]  

(3)

F is a parameter, representing that the chicken will follow his mother to find food. Considering individual differences, F of each chicken will be selected randomly between [0, 2], \( m \) represents the position of the mother of the first chicken.

The traditional CSO algorithm has the disadvantages of low population diversity and easy to fall into local extremum. In this paper, the weighting coefficient is used to improve the optimization performance and efficiency of the algorithm.
3. CSO optimization based on adaptive inertia weight coefficient

In the CSO algorithm, the update of the position of the chicken only follows the position of the chicken mother, and there is no movement to the optimal orientation of the individual rooster in the algorithm, so that it is easy to cause the individual to fall into local optimum to some extent and the overall efficiency of the algorithm. Lowering, the chicks in the CSO and the particles in the particle swarm algorithm have similarities. The particles in the particle swarm algorithm obtain the optimal solution position locally and obtain the optimal solution position, which is equivalent to the chicken getting the local position next to the hen and the optimal position in the group led by the entire rooster. Therefore, this paper updates the formula 6 by referring to the concept of learning factors in the particle swarm optimization algorithm:

\[ x_{i,j}(t+1) = x_{i,j}(t) + \lambda_1(x_{m,j}(t) - x_{i,j}(t)) + \lambda_2(x_{r,j}(t) - x_{i,j}(t)) \]  

(7)

In formula 6, \( m \) denotes the individual in the subgroup following the corresponding hen, \( r \) is the individual of the cock corresponding to the chick in the subgroup \( \lambda_1, \lambda_2 \) respectively represent two learning factors, \( \lambda_1 \) represents the degree factor of learning the chick to the hen, \( \lambda_2 \) represents the degree factor that the chick learns from the rooster.

The values of \( \lambda_1 \) and \( \lambda_2 \) can determine the degree of learning from hens and roosters to a certain extent. The setting of \( \lambda_1 \) and \( \lambda_2 \) values can easily lead to premature convergence or divergence of the algorithm. Therefore, it is necessary to limit the values of \( \lambda_1 \) and \( \lambda_2 \), \( \lambda_{\text{max}} \) and \( \lambda_{\text{min}} \) are expressed as the maximum and minimum values of learning factors respectively. \( t_{\text{current}} \) is the current iteration times. \( t_{\text{max}} \) and \( t_{\text{min}} \) represent the maximum and minimum number of times set respectively. The purpose of this setting is to enable chickens to learn better from rooster and hens.

\[
\lambda_1 = \begin{cases} 
\lambda_1 \times \frac{t_{\text{max}} - t_{\text{current}}}{t_{\text{max}}}, & \lambda_1 > \lambda_{\text{min}} \\
\lambda_{\text{min}}, & \lambda_1 \leq \lambda_{\text{min}} 
\end{cases}
\]  

(8)

\[
\lambda_2 = \begin{cases} 
\lambda_2 \times \frac{t_{\text{max}} + t_{\text{current}}}{t_{\text{max}}}, & \lambda_2 < \lambda_{\text{min}} \\
\lambda_{\text{max}}, & \lambda_2 \geq \lambda_{\text{max}} 
\end{cases}
\]  

(9)

4. Cross mutation based on genetic algorithm

In CSO algorithm, roosters, hens and chicks will complete the final optimization process through their own fitness value ranking and mutual learning. In order to solve the local convergence phenomenon caused by population identity and premature with the increase of iteration times, this paper introduces the idea of genetic algorithm to carry out crossover and mutation operations respectively.

The specific process is as follows:

a. The roosters were sorted according to the fitness value, the crossover probability \( P_c = 0.5 \), and the range value of hybridization pool was determined;

b. Select the position and velocity information of the first 10 particles, and then pick out particles 1 and 2 to cross generate new particles

\( (P_c: \text{ probability of generating new particles, whose value is a random number between 0 and 1}) \);

c. Updating the position and speed of the offspring and calculating the fitness value to judge whether the position and speed of the offspring replace the position and speed of the parent;

d. Let the mutation probability \( P_m = 0.05 \), determine the size of the rooster mutation pool, select the mutation particle 3, calculate the fitness value and judge whether to replace the parent position.

5. Experiment

In this paper, we improved the CSO algorithm, and naming ICSO. To test the effectiveness of ICSO algorithm, we will calculate the standard test function in Table 1, comparing the superiority between CSO and ICSO. In this paper, we consider the following evaluation criteria to compare the two
algorithms: a) Optimization accuracy, the optimal fitness value of the search under the calculation of the fixed number of functions. b) Stability, the variance of the fitness obtained after the algorithm is run multiple times independently.

| Function | Expression | Bounds   | Optimal value |
|----------|------------|----------|---------------|
| Sphere   | $f_1(x) = \sum_{j=1}^{D} x_j^2$ | [-5.12, 5.12] | 0 |
| Rosenbroken | $f_2(x) = \sum_{j=1}^{D} [100(x_j - x_j^2) + (1 - x_j^2)]$ | [-30, 30] | 0 |
| Rastrigin | $f_3(x) = 10D + \sum_{j=1}^{D} [x_j^2 - 10\cos(2\pi x_j)]$ | [-5.12, 5.12] | 0 |
| Ackley   | $f_4(x) = -20e^{-0.2\sqrt{\frac{x_1^2 + x_2^2}{2}}} - e^x + 20 + e$ | [-32, 32] | 0 |

Set chicken flock size $N=100$, $N_R=0.2N$, $N_I=0.6N$, $N_c=0.1N$, $N_M=N_I/6$, $G=10$, maximum iteration number $n_{max}=100$. In the CSO algorithm, $F_{rand}(0.4, 1)$, in the improved CSO algorithm, $=0.9$, $=0.1$, $=0.8$, $=0.1$. The two algorithms independently run each test function 30 times, and find the optimal value, the average value and the standard deviation of the 30 running results.

When dimension D=10, the results are shown in Table 2.

| Function | Bounds   | Algorithm | Optimal value | Worst value | Average value | Standard deviation |
|----------|----------|-----------|---------------|-------------|---------------|-------------------|
| $f_1$    | [-5.12, 5.12] | CSO       | 2.6991*10^{-129} | 2.3743*10^{-123} | 1.5892*10^{-124} | 6.1315*10^{-124} |
|          |          | ICSO      | 1.5537*10^{-137} | 21.3547     | 6.148*10^{-132} | 1.7998*10^{-131}  |
| $f_2$    | [-30, 30] | CSO       | 6.2851         | 7.2040      | 6.8302        | 0.29766          |
|          |          | ICSO      | 6.2527         | 7.2447      | 6.8292        | 0.29515          |
| $f_3$    | [-5.12, 5.12] | CSO       | 0             | 0           | 0             | 0                |
|          |          | ICSO      | 0             | 0           | 0             | 0                |
| $f_4$    | [-32, 32] | CSO       | 8.8818*10^{-16} | 2.6645*10^{-15} | 6.5133*10^{-16} | 9.1731*10^{-16} |
|          |          | ICSO      | 0             | 0           | 0             | 0                |

When dimension D=100, the results are shown in Table 3.

| Function | Bounds   | Algorithm | Optimal value | Worst value | Average value | Standard deviation |
|----------|----------|-----------|---------------|-------------|---------------|-------------------|
| $f_1$    | [-5.12, 5.12] | CSO       | 5.0715*10^{-7} | 155.3193    | 29.6689       | 50.0285          |
|          |          | ICSO      | 8.7393*10^{-20} | 21.3547 | 1.8675        | 5.6424           |
| $f_2$    | [-30, 30] | CSO       | 1.5143*10^{6} | 1.4292*10^{7} | 6.3296*10^{6} | 3.4858*10^{6}   |
|          |          | ICSO      | 6.4736         | 104.739     | 72.3293       | 45.1791          |
| $f_3$    | [-5.12, 5.12] | CSO       | 3.9847*10^{-9} | 1.2789*10^{-5} | 3.2610*10^{-6} | 3.8657*10^{-6} |
|          |          | ICSO      | 0             | 3.908*10^{-14} | 2.6053*10^{-15} | 1.009*10^{-14}  |
| $f_4$    | [-32, 32] | CSO       | 9.1803*10^{-5} | 9.8348     | 5.2904        | 3.4052           |
|          |          | ICSO      | 3.5682*10^{-22} | 4.4461*10^{-19} | 1.6366*10^{-19} | 1.0603*10^{-19} |

When dimension D=200, the results are shown in Table 4.
Table 4 Test function optimization value (D=200)

| Function | Bounds     | Algorithm | Optimal value | Worst value | Average value | Standard deviation |
|----------|------------|-----------|---------------|-------------|---------------|--------------------|
|          |            | CSO       | 2.1095×10^1  | 2.4692×10^4 | 3.4833×10^3  | 4.632×10^3        |
| f_1      | [-5.12, 5.12] | ICSO      | 1.4360×10^-5 | 3.7524×10^0 | 5.5770×10^-1 | 8.8122×10^-1     |
|          |            | CSO       | 1.9096×10^7  | 5.3231×10^7 | 3.2325×10^7  | 9.3689×10^6       |
|          |            | ICSO      | 4.8765×10^-3 | 5.8352×10^1 | 6.9333×10^-1 | 1.3467×10^-1     |
| f_2      | [-30, 30]  | CSO       | 8.2850×10^-1 | 2.3962×10^2 | 5.0030×10^-1 | 5.7730×10^-1     |
|          |            | ICSO      | 1.1357×10^-3 | 4.5916×10^-1 | 7.1891×10^-2 | 1.1719×10^-1     |
| f_3      | [-600, 600] | CSO       | 8.7751×10^0  | 1.2082×10^1 | 1.0358×10^1  | 6.9711×10^-1     |
|          |            | ICSO      | 1.2562×10^-3 | 3.8869×10^-1 | 5.1326×10^-2 | 7.9103×10^-2     |
| f_4      | [-32, 32]  | CSO       | 1.8017×10^-2 | 8.1821×10^-1 | 2.7221×10^-1 | 2.6551×10^-1     |
|          |            | ICSO      | 5.2043×10^-4 | 1.3701×10^0 | 1.9970×10^-1 | 3.4581×10^-1     |

In Table 1, the global optimality of the four test functions is 0. In Table 2 and Table 3, when D=10, the improved CSO is effective and stable in f_1, f_2, and f_4. The performance is better than the result of CSO; when D=100, the improved CSO is better than CSO in the solution and stability of f_1, f_2, f_3 and f_4. It is concluded that the improved algorithm can improve the precision of optimization to a certain extent, and reduce the occurrence of individual optimal problems in chicks to a certain extent. The stability of the fitness function is more stable than the original algorithm.

6. Conclusion

In the improved CSO algorithm, the adaptive inertia weight factor is added to the hen position formula. Considering the influence of rooster in the same subgroup on the chick formula, the learning factor for chicks to learn from rooster is increased. The cross mutation enriches the population diversity, improves the population diversity, makes the particles trapped in the local optimal jump out of the local minimum, and finally calculates the individual fitness value to judge whether to replace the parent position and speed. When dealing with high dimensions, the algorithm can also jump out of local stagnation for global optimization.

The optimized CSO algorithm is tested by five test functions. The simulation results show that the improved CSO algorithm has fast convergence speed and strong stability, avoids the problem of premature convergence, and greatly improves the optimization accuracy compared with the original algorithm.

References

[1] Mashwani W, Haider R, Belhaouari S.A Multiswarm Intelligence Algorithm for Expensive Bound Constrained Optimization Problems[J]. Complexity, 2021, 2021(4):1-18.
[2] Meng X, Yu L, Gao X, et al. A New Bio-inspired Algorithm: Chicken Swarm Optimization[C]//International Conference in Swarm Intelligence. Springer International Publishing, 2014.
[3] Jiang S, Mashdoor S, Parvin H, et al. An adaptive location-aware swarm intelligence optimization algorithm[J]. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2021, 29(2): 249-279.
[4] Cui D. Projection pursuit model for evaluation of flood and drought disasters based on chicken swarm optimization algorithm[J]. Advances in ence & Technology of Water Resources, 2016.
[5] Souaille T, Petiot J, Lagrange M, et al. EXTRACTING DESIGN RECOMMENDATIONS FROM INTERACTIVE GENETIC ALGORITHM EXPERIMENTS: APPLICATION TO THE DESIGN OF SOUNDS FOR ELECTRIC VEHICLES[J]. 2021.
[6] Yedjou D. Application of the Genetic Algorithm to the Rule Extraction Problem[M]. 2021.
[7] HUANG G B, WANG D H, LAN Y. Extreme learning machines: a survey [J]. International journal of machine learning and cybernetics, 2011, 2(2): 107-122.