A discrete chicken swarm optimization for traveling salesman problem

Yuanjie Liu¹, Qiang Liu and Zhi Tang

School of Information and Control Engineering, Liaoning Petrochemical University, Fushun, 113001, China

¹Corresponding author’s e-mail: 2641913962@qq.com

Abstract. Traveling Salesman Problem (TSP) is a typical combinatorial optimization problem, and it is a NP-hard problem. The total number of routes increases exponentially with the number of cities, so it is great significance to design an effective algorithm to find the optimal solution accurately. Chicken Swarm Optimization (CSO) is a new intelligent optimization algorithm, which is mainly proposed for continuous problems. It has the advantages of fast convergence speed and high convergence accuracy. This paper proposed a Discrete Chicken Swarm Optimization (DCSO) for TSP. The CSO is discretized by introducing the methods of swap, order crossover and reverse order mutation, where the search space of the solution is enlarged, and the diversity of the solution is increased. The typical TSP models are simulated and compared with the Basic Ant Colony Optimization and Genetic Algorithm to verify the feasibility of the presented method.

1. Introduction
Since Menger proposed Traveling Salesman Problem (TSP) in 1932, it has attracted many researchers' interest in various fields. It is a typical combinatorial optimization problem, but it has not been completely solved. In many fields, complex optimization problems can be transformed into TSP problems by simple description, such as computer networking, transportation, circuit design, workshop scheduling problems, so the solution of TSP problem has important value in theory and practical application.

Traditional methods such as exhaustive method, branch and bound method and dynamic programming method are often used to solve small TSP problems. However, when solving large-scale TSP problems, the complexity is increasing, and the traditional methods are less and less able to find the optimal solution in the huge search space, so it is necessary to find new algorithms. At present, the main methods to solve large TSP are Ant Colony Optimization (ACO) [1-3], Genetic Algorithm (GA) [4-6], Particle Swarm Optimization (PSO) [7], Simulated Annealing (SA) [8,9], and some hybrid intelligent optimization algorithms [10,11]. Solving TSP is also used to verify the effectiveness of the algorithm.

Chicken Swarm Optimization (CSO) is a kind of swarm intelligence optimization algorithm proposed by Meng et al. [12] at the Fifth International Conference in Swarm Intelligence. In the algorithm, the chicken swarm is divided into roosters, hens and chicks by simulating the hierarchy order and behavior of the chicken swarm. By making different optimization strategies for different swarm, the algorithm searches randomly in the global scope. Compared with PSO, Differential Evolution (DE) and Bat Algorithm (BA) [13], it has the advantages of fast convergence speed and high convergence speed. Since the CSO was proposed, it has attracted the attention and research of many scholars in related fields,
and has been successfully applied to engineering optimization design. Hong et al. [14] applied the CSO to multi-classifier coefficient optimization, which reduced the error rate of multi-classifier and shortened the training time. Irsalind et al. [15] proposed a clustering analysis method based on CSO, which has better clustering performance compared with intelligent optimization algorithms such as PSO and GA. Wang et al. [16] applied the CSO to the distribution network reconstruction to improve the search efficiency. Dong et al. [17] proposed a WS+ multi-target location algorithm based on the compressed perception of discrete CSO, which realized the accurate reconstruction of the multi-target location information with unknown sparsity. Zhou et al. [18] proposed a discrete CSO to solve the 0-1 knapsack problem, which improves the quality of understanding and convergence speed. Ye et al. [19] proposed an improved CSO, which uses the strategy of exchange operator and exchange sequence to optimize the discrete path. Meanwhile, it combines the strategy of 2-opt operator, self-exploration and simulated annealing operator to improve the solution quality of the algorithm. Chebihi et al. [20] proposed a new discrete CSO to solve the symmetric TSP by redefining the basic different operators and operations of CSO algorithm, and the quality of the solution is improved by 2-opt local search. These applications show that the algorithm has a good application prospect. However, there are few researches on the application of discrete optimization. Compared with [19] and [20], this paper adopts a new method to discretize CSO. By introducing swap, order crossover and reverse order mutation, a new Discrete Chicken Swarm Optimization (DCSO) is proposed to solve TSP. Through the simulation experiment, the feasibility of the presented method is verified by comparing with Basic Ant Colony Optimization (BACO) and GA.

2. Basic chicken swarm optimization
The CSO divides the group by simulating the hierarchy order and behavior of the chicken swarm. Different kinds of chickens have different responsibilities. Follow these principles:

1. There are many groups in a chicken swarm, each of which consists of a rooster, several hens and some chicks.
2. Through the fitness to establish a hierarchy order and distinguish different types of chickens. In the chicken swarm, the individuals with the best fitness value are the roosters and each rooster is the head of a group. The individuals with the worst fitness value are the chicks, and the remaining individuals are the hens. The dominance relationship between the hen and the rooster, and the mother-child relationship between the hen and the chick are all randomly established.
3. In each group, hierarchy order, dominance relationship and mother-child relationship remain unchanged as long as they are established, and will be updated only when the update cycle (G) is reached.
4. The chickens in each group follow the roosters to find food, which can prevent other individuals from grabbing their own food. They can also randomly steal the food found by other individuals, and each chick follows their mother to find food. The dominant chickens have a good competitive advantage, they find food before other individuals.

3. Discrete chicken swarm optimization for traveling salesman problem
TSP is a typical NP-problem, so it is of great significance to find an effective solution. CSO has been successfully applied to continuous optimization problems, but the application of discrete problems is rare. In the optimization algorithm, swap, order crossover and reverse order mutation are all feasible operators to deal with discrete problems. In this paper, swap, order crossover and reverse order mutation are introduced into CSO to solve discrete optimization problem.

3.1. TSP mathematical model
The TSP problem is to know the distance between all cities. The traveling salesman will visit each city only once from a certain city, and finally return to the starting city to find the shortest route to visit all cities.
The collection of known cities is \( \{c_1, c_2, \ldots, c_n\} \). The distance between the two cities \( i \) and \( j \) is \( d(c_i, c_j) \). TSP problem to find a permutation \( x = (c_1, c_2, \ldots, c_n) \) with the minimum path length.

\[
 f(x) = \sum_{i=1}^{n-1} d(c_i, c_{i+1}) + d(c_1, c_n) 
\]  

(1)

It is assumed that each city is interlinked, and the distance between adjacent cities is independent of the path. Each city passes only once, and the minimum value of the objective function \( f(x) \) is solved.

3.2. Individual encoding
For the TSP problem of \( n \) cities, the integer coding method is adopted, and the position of each chicken corresponds to a path. Suppose \( x = (c_{i_1}, c_{i_2}, \ldots, c_{i_M}) \) is a feasible solution of the objective function, which means that the route visited by the traveling salesman is \( c_1 \rightarrow c_{i_2} \rightarrow \cdots \rightarrow c_{i_M} \rightarrow c_{i_1} \). \( c_{i_j} = 1, 2, \cdots, n \), \( i = 1, 2, \cdots, M \), \( j = 1, 2, \cdots, n \). \( M \) is the size of the swarm. \( n \) is the number of cities. \( c_{i_j} \) represents the \( j \)-dimension element of the \( i \)-th chicken.

3.3. Swap
Swap [21] is introduced into the CSO to solve the discrete optimization problem. The sequence of TSP problems is \( x = (c_1, c_2, \ldots, c_n) \). The swap \( x = (i_1, i_2) \) is defined as \( c_{i_1} \rightarrow c_i_{i_2} \rightarrow \cdots \rightarrow c_{i_M} \rightarrow c_{i_1} \). \( c_{i_j} = 1, 2, \cdots, n \). \( i = 1, 2, \cdots, M \). \( j = 1, 2, \cdots, n \). \( M \) is the size of the swarm. \( n \) is the number of cities. \( c_{i_j} \) represents the \( j \)-dimension element of the \( i \)-th chicken.

\[
 x' = x + xo(i_1, i_2) \text{ is a new solution of } x \text{ after } xo = (i_1, i_2) \text{ operation. For example:}
\]
\[
 x = 5, 7, 3, 2, 4, 1, 6, 8
\]
\[
 x' = x + xo(2, 3) = 5, 3, 7, 2, 4, 1, 6, 8
\]

3.4. Order crossover
Order crossover [22] is introduced into the CSO to solve the discrete optimization problem. Two individuals were selected for order crossover. Firstly, two tangent points \( x \) and \( y \) are randomly selected, and the middle part of the two tangent points is exchanged. Secondly, the original sequence is listed from the first position after the tangent point \( y \), and the existing sequence number is removed. Finally, the corresponding sequence number is filled in from the first position after \( y \). For example:

Paternal generation \( P: 2 \mid 3 \ 4 \ 5 \mid 6 \ 8 \ 7 \)

Paternal generation \( Q: 8 \ 3 \mid 1 \ 2 \ 5 \mid 7 \ 6 \ 4 \)

After order crossover, the new sequence is as follows.

Filial generation \( P: 3 \ 4 \mid 1 \ 2 \ 5 \mid 6 \ 8 \ 7 \)

Filial generation \( Q: 1 \ 2 \mid 3 \ 4 \ 5 \mid 7 \ 6 \ 8 \)

3.5. Reverse order mutation
Reverse order mutation [22] is introduced into the CSO to solve the discrete optimization problem. In the individual coding, two tangent points \( x \) and \( y \) are randomly selected, and the sequence numbers between them are arranged in reverse order to form a new sequence number. For example:

\( T = 4 \ 6 \ 1 \ 7 \ 3 \ 5 \ 2 \ 8 \)

After reverse order mutation, the new sequence is as follows.

\( T1 = 4 \ 6 \ 1 \ 7 \ 3 \ 5 \ 2 \ 8 \)

3.6. Discrete chicken swarm optimization
In order to expand the search space of the solution, the \( i \)-th individual is operated by a swap and the new individual position is calculated. If the fitness of the new position is smaller than that of the original position, the individual position will be updated.

There are different update formulas due to the different searching ability of roosters, hens and chicks. The rooster with good adaptability obtains food before other individuals, searches for food in a larger range, and realizes global search. The position update of roosters is affected by the positions of other roosters randomly selected. The update formula of rooster position of DCSO is as follows,
where $x_{ij}^t (i \in [1, \ldots, N], j \in [1, \ldots, D])$ represents the position of the $i$th chicken in time step $t$, looking for food in $D$-dimensional space. $k \in [1, N], k \neq i$, $k$ is the rooster different from the $i$th, $\otimes$ represents order crossover operation, indicating order crossover operation of $i$th rooster and $k$th rooster.

The search ability of hens is worse than that of roosters. They search for food around roosters. The position update is affected by the position of other roosters. Due to the hens can steal the food of chickens with better adaptability than themselves, there is a competitive relationship with other chickens, and the position update of hens is affected by other roosters and hens. The update formula of hen position of DCSO is as follows,

$$x_{ij}^{t+1} = (x_{ij}^t \otimes x_{r1,j}^t) \otimes x_{r2,j}^t$$

where $r1 \in [1, N], r2 \in [1, N], r2$ is the randomly selected rooster or hen in the chicken swarm, and $r1 \neq r2$.

Chicks have the worst search ability. They follow their mother to search for food. The scope of search is small. The update of chicks is affected by their mother's position. The update formula of chick position of DCSO is as follows,

$$x_{ij}^{t+1} = x_{ij}^t \otimes x_{m,j}^t$$

where $x_{m,j}^t$ is the position of the mother of the $i$th chick.

After different order crossover operations on different chickens, new individual positions are calculated. If the fitness of the new position is smaller than that of the original position, the individual position will be updated.

In the process of optimization, in order to increase the diversity of individuals and avoid falling into local optimum, the reverse order mutation operation is introduced, and the new individual position is calculated. If the fitness of the new position is smaller than that of the original position, the individual position will be updated.

The pseudo-code of the DCSO is as follows.

**Discrete Chicken Swarm Optimization**

Initialize a population of $N$ chickens and define relevant parameters.

Calculate individual fitness value $f(P_i)$, $t = 1$.

While ($t < \text{Max\_Generation}$)

If ($t \% G == 0$)

Establish a hierarchy order in the swarm.

End if

For $i = 1: N$

//Swap

New results are calculated by swap.

Update the new solutions if they are better than the previous one.

//Order crossover

If $i == \text{rooster}$

$$x_{ij}^{t+1} = x_{ij}^t \otimes x_{k,j}^t$$

Update the new solutions if they are better than the previous one.

End if

If $i == \text{hen}$

$$x_{ij}^{t+1} = (x_{ij}^t \otimes x_{r1,j}^t) \otimes x_{r2,j}^t$$

Update the new solutions if they are better than the previous one.

End if

If $i == \text{chick}$

$$x_{ij}^{t+1} = x_{ij}^t \otimes x_{m,j}^t$$

Update the new solutions if they are better than the previous one.
if //Reverse order mutation
    New results are calculated by reverse order mutation.
    Update the new solutions if they are better than the previous one.
End for

t = t+1
End While

4. Simulation experiment results and analysis
In order to test the optimization ability of the DCSO, this paper selects the typical city model of TSP problem to test. Compared with BACO and GA, the feasibility and effectiveness of DCSO are verified. Test computer hardware environment: the processor is Intel (R) core (TM) i5-1035G1CPU@1.00GHz. The simulation experiment is carried out in MATLAB 2018a. The algorithm parameters are set as follows: G = 5. Roosters, hens and mother hens accounted for 20%, 60% and 10% of the population respectively. Each group of experiments is tested independently, and the relevant test results are recorded. The known optimal solution of the city model in TSPLIB is shown in Table 1. It can be seen from [23] that the optimal solution of Eil50 is 427.8552.

| City model | Ulysses16 | Oliver30 | Eil50 | Pr76 | Pr107 |
|------------|-----------|---------|-------|------|-------|
| Known optimal solution | 73.9876 | 423.7406 | 427.8552 | 108159 | 44303 |

In order to verify the effectiveness of DCSO for TSP, Ulysses16, Oliver30, Eil50, Pr76 and Pr107 are selected for testing. The test results of the three algorithms are shown in Table 2. Ulysses16 and Oliver30 are small-scale problems in TSP. It takes 1.6 seconds for DCSO to solve Ulysses16, which converges in about 10 generations, and 3.5 seconds to solve Oliver30, which converges in about 35 generations. According to the data in Table 2, the optimization result of DCSO is better than that of BACO and GA, and the average value of 30 experiments is also the minimum. It shows that DCSO can successfully solve the small-scale TSP and find the shortest path, which verifies the feasibility of the presented method.

| TSP model | algorithm | Max      | Min      | Average value | Standard deviation |
|-----------|-----------|----------|----------|---------------|--------------------|
| Ulysses16 | BACO      | 74.6287  | 74.1087  | 74.5002       | 0.2020             |
|           | GA        | 74.6657  | 73.9876  | 74.1346       | 0.1785             |
|           | DCSO      | 73.9998  | 73.9876  | 73.9888       | 0.0037             |
| Oliver30  | BACO      | 429.7853 | 423.9117 | 426.5655      | 1.2484             |
|           | GA        | 491.4149 | 423.7406 | 452.7812      | 20.5629            |
|           | DCSO      | 428.4642 | 423.7406 | 424.0440      | 0.9209             |
| Eil50     | BACO      | 451.9704 | 439.6038 | 447.3958      | 2.7965             |
|           | GA        | 524.0288 | 444.1869 | 476.5415      | 21.0163            |
|           | DCSO      | 442.7896 | 430.5125 | 436.8592      | 3.7357             |
| Pr76      | BACO      | 121967   | 116530   | 119918        | 1297               |
|           | GA        | 158135   | 129246   | 141205        | 8016               |
|           | DCSO      | 114028   | 108396   | 111097        | 1490               |
| Pr107     | BACO      | 47277    | 46327    | 46720         | 222                |
|           | GA        | 81788    | 68320    | 76071         | 3474               |
|           | DCSO      | 46372    | 44385    | 44873         | 484                |

In order to further verify the optimization ability of DCSO, the model with more cities is selected to test. With the increase of the number of cities, the difficulty of solving the problem increases gradually.
It takes 26 seconds, 78 seconds and 132 seconds respectively for DCSO to solve Eil50, Pr76 and Pr107, which converges about 80, 100 and 115 generations respectively. With the increase of the number of cities, the results of BACO and GA are much different from the current known optimal solution. The results of DCSO are the closest to the known solutions, and the average value of 30 tests is the smallest, which indicates that the solution quality of DCSO is higher. The effectiveness of the presented method is verified.

5. Conclusion
In order to solve TSP problem, a new DCSO algorithm is proposed. Firstly, through the analysis of TSP problem, the mathematical model is established. Secondly, CSO is discretized by introducing swap, order crossover and reverse order mutation. Finally, the typical TSP models are selected for testing. Meanwhile, compared with BACO and GA, the feasibility and effectiveness of the algorithm are verified. From the test results, the algorithm has great potential and broad application prospects. This paper can provide some references for CSO to deal with discrete problems. In the future research, the solution quality of DCSO will be improved, and it will be combined with practical application, such as solving the pipe sequence optimization of aero-engine.

References
[1] Huang L, Zhou C and Wang K P 2003 Hybrid ant colony algorithm for traveling salesman problem Progress in natural science 13 295-299
[2] Peng X, Zhou Y R and Xu G 2016 Approximation performance of ant colony optimization for the TSP problem International journal of computer mathematics 93(10) 1683-1694
[3] Yang Y J, An J X and Han H Y 2014 Ant colony algorithm based on semantic relations and its applications Applied mechanics and materials 644 2076-2080
[4] Kaabi J and Harrath Y 2019 Permutation rules and genetic algorithm to solve the traveling salesman problem Arab journal of basic and applied sciences 26(1) 283-291
[5] Ahmed Z H 2014 Improved genetic algorithms for the travelling salesman problem Int. J. of process management and benchmarking 4(1) 109-124
[6] Wang J Q, Ersoy OK, He M Y and Wang F L 2016 Multi-offspring genetic algorithm and its application to the traveling salesman problem Applied soft computing 43 415-423
[7] Sriborikit J and Tuwanut P 2015 An improvement particle swarm optimization for travelling salesman problem with the mutation operator Applied mechanics and materials 781 527-530
[8] Ezugwu AES, Adewumi AO and Frîncu ME 2017 Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem Expert systems with applications 77 189-210
[9] Wang C Y, Lin M, Zhong Y W and Zhang H 2016 Swarm simulated annealing algorithm with knowledge-based sampling for travelling salesman problem Int. J. of intelligent systems technologies and applications 15(1) 74-94
[10] Saenphon T, Phimoltaires S and Lursinsap C 2014 Combining new fast opposite gradient search with ant colony optimization for solving travelling salesman problem Engineering applications of artificial intelligence 35 324-334
[11] Huo L, Jiang B and Ning T 2015 A new algorithm for solving TSP and its applications Applied mechanics and materials 713 1504-1508
[12] Meng X B, Lin Y, Gao X Z and Zhang H Z 2014 A new bio-inspired algorithm: Chicken swarm optimization International conference in swarm intelligence 8794 86-94
[13] Das S and Suganthan P N 2011 Differential evolution: a survey of the state-of-the art IEEE Transactions on evolutionary computation 15(1) 4-31
[14] Hong Y and Yu F Q 2017 Improved chicken swarm optimization and its application in coefficients optimization of multi-classifier Computer engineering and application 53(9) 158-161
[15] Irsalind N, Yanto ITR, Chiroma H, Chiroma H and Herawan T 2016 A framework of clustering
based on chicken swarm optimization *International conference on soft computing and data mining* 549 336-343

[16] Wang X C, Hu H M and Liu L 2016 Distribution network reconfiguration based on chicken swarm optimization algorithm *Electrotechnics electric* 03 20-24

[17] Dong Y Q and Wang H 2017 Multiple target localization WSNs via CS reconstruction method based on discrete CSO algorithm *Computer and modernization* 12 23-27

[18] Zhou Y and Pan D Z 2019 The application of the discrete chicken swarm optimization (DCSO) in 0-1 knapsack problem *Intelligent computer and application* 9 (01) 98-103

[19] Ye H, Fu Q, Ye J and Zhong C 2019 An Improved Chicken Swarm Optimization for TSP. *In International Conference on Applications and Techniques in Cyber Security and Intelligence* Springer Cham 211-220

[20] Chebihi F, Riffi ME, Agharghor A, Semlali SCB and Haily A 2021 Improved Chicken Swarm Optimization Algorithm to Solve the Travelling Salesman Problem *Indonesian Journal of Electrical Engineering and Computer Science* 12(3) 1054-1062

[21] He F G 2014 Research on information applied technology with swarm intelligence for the TSP problem *Advanced Materials Research* 886 584-588

[22] Liu Z Y, Zhou J, Zhang L, Zhang J X, Zhang Q and Sha R 2018 Research on TSP problem based on crossover and mutation combination *Computer and modernization* 03 54-59

[23] Tao L H, Ma Z N, Shi P T and Wang R F 2019 Dynamic ant colony genetic algorithm based on TSP *Machinery Design & Manufacture* 12 147-149+154