Logistics Route Optimization Based on Improved Particle Swarm Optimization

Zunhai Gao¹ and Hongxia Lu²,*

¹College of Artificial Intelligence, Nanchang Institute of Science and Technology, Nanchang, China
²School of Management, Wuhan Donghu University, Wuhan, China

*Corresponding author email: luhx@wdu.edu.cn

Abstract. An improved particle swarm optimization (PSO) algorithm is presented by dynamically adjusting the inertia weight in the iterative process of PSO, and it is used to solve the problem of logistics route optimization. This algorithm can not only improve the convergence speed, but also avoid falling into local optimum. In the process of improving the standard algorithm, two methods are proposed to adjust the inertia weight value according to the number of iterations. One is piecewise linear decreasing, another is linear decreasing. The results show that linear decline is better than piecewise linear decline to achieve the purpose of optimization, which is more conducive to accelerate the convergence rate and enhance the ability of optimization. Through the simulation experiment of the specific vehicle routing optimization problem, the results show that after the improvement, the optimization performance is enhanced, the optimization speed is accelerated, and the complexity is not increased, which greatly improves the performance of the original algorithm.

Keywords: Logistics route optimization; Improved PSO algorithm; Inertia weight.

1. Introduction

The basic idea of particle swarm optimization (PSO) comes from the research and simulation of birds’ foraging behaviour. The algorithm is simple in concept, easy to implement, and has strong global search ability. It is an effective to solve the distribution route optimization problem, but it is easy to fall into the local optimal solution. In view of this, many improved algorithms have proposed [1-3]. The purpose of logistics path optimization is to find a reasonable distribution scheme with low cost and high speed among many distribution schemes. Once the problem was put forward, it soon attracted the attention and research of experts and transportation policy makers. It is a well-known NP hard problem to optimize logistics distribution routes, including vehicle routing problem (VRP) and travelling salesman problem (TSP). VRP Problem is defined as: some limited vehicle sets, some customer points, each customer point has different goods needs, under certain constraints, it is required to arrange the appropriate distribution scheme, complete the needs of each customer, and achieve a certain goal (such as the lowest cost, shortest route, fastest speed, etc.). It is not easy to find the exact solution of VRP Problem. Now the effective algorithm to solve the path optimization is in constant progress, and there are many discoveries in its theoretical part [6-11], but there are still some areas in urgent need of improvement and improvement in the operation time and the quality of the optimal solution. So we present an improved particle swarm optimization (PSO) algorithm by dynamically adjusting the inertia weight in the iterative process of PSO, and it is used to solve the problem of logistics route optimization.
2. Particle Swarm Optimization and Its Improvement

2.1. Standard Particle Swarm Optimization

In standard particle swarm optimization, the position of each individual is considered as the feasible solution of the problem. In the solution space, individuals dynamically change their own motion direction and speed by virtue of their own flight experience and other individuals in the space, until they find the optimal position, that is, the optimal solution of the problem. With the increase of iterations, the velocity and position of particles are also changing.

There are some parameters in PSO: balance the global and local search weight $w$, algorithm parameters $c_1$, $c_2$, number of individuals $N$, speed limit, search times and so on. Now we set up the following: in the search area of $D$ dimension, there are $N$ individuals in the population, and their positions are as follows $x_1, x_2, \cdots, x_N$, the speed is $v_1, v_2, \cdots, v_N$. Then, the location and velocity of each individual are initialized randomly, which are all $D$ dimensions. For example, the position of the $j$-th particle is $x_j = (x_{j1}, x_{j2}, \cdots, x_{jd})$. The velocity of the $j$-th particle is $v_j = (v_{j1}, v_{j2}, \cdots, v_{jd})$.

At the same time, the local optimal value of each individual and the global optimal value of all individuals are changed according to the fitness value. Let the best location (local best) found by the individual $i$-th be $p_i = (p_{i1}, p_{i2}, \cdots, p_{id})$. The best location for all individuals is $p = (p_1, p_2, \cdots, p_N)$.
The position and velocity of the $j$-th component of particle $i$ are updated according to formulas (1) and (2), respectively

$$v_{ij}(t + 1) = w v_{ij}(t) + c_1 r_1 (p_{ij}(t) - x_{ij}(t)) + c_2 r_2 (p(t) - x_{ij}(t))$$

$$x_{ij}(t + 1) = x_{ij}(t) + v_{ij}(t + 1)$$

Where $i$ is the number of the current evolution (i.e. the number of iterations), $i = 1, 2, \cdots, N$, indicates that the location and rate of all individuals are updated. $c_1$ and $c_2$ are learning factors (the larger $c_1$ is, the greater the experience of the representative particles has on the individual, the greater $c_2$, the greater the experience of the representative group has on each particle, all of which are greater than 0). $r_1$ and $r_2$ are random numbers distributed evenly between 0 and 1. $i$ determines the effect of the original rate on the current rate, which is of great significance to balance global search and local searching ability.

2.2. Improvement of Particle Swarm Optimization

This paper will dynamically change the inertia weight according to the number of search times, so as to achieve the purpose of optimizing the algorithm. At the beginning of the iteration, taking a larger inertia weight makes the speed between particles play a greater role, increases the individuality and diversity of particles, expands the search range of particles, increases the iteration efficiency, and avoids falling into the local optimum due to the aggregation of particles. With the increase of the number of iterations, the particles are gradually close to the optimal position. At this time, the smaller inertia weight is taken to increase the local search ability of particles. Therefore, this paper proposes a method to dynamically adjust the inertia weight according to the number of iterations, which can not only enhance its optimization performance, but also prevent it from entering the local optimum. In the whole iteration process, the inertia weight can be changed from time to time. There are two methods to change the inertia weight according to the number of iterations ($T$ representing the total number of iterations):

**Improved algorithm 1:**

$$w = w - 100/T$$

In this way, formula (3) is called to update the value of inertia weight every 100 iterations, that is, the value is reduced by 100/total iterations. In the process of individual search, the inertia weight is reduced by segments.

**Improved algorithm 2:**

$$w = w - 1/T$$

Every iteration, the formula (4) is used to update the value of inertia weight, that is, reduces it by
100/total iterations. With the increase of search times, the inertia weight decreases linearly. In the whole optimization process, the inertia weight of the above two improved algorithms are reduced by one unit. Both methods gradually reduce the value of inertia weight in the process of increasing the number of iterations, but the improved method is piecewise linear decreasing, and the improved method is linear decreasing.

3. Vehicle Routing Optimization Problem

There are $K$ vehicles participating in the delivery of goods. The maximum length of each vehicle is $D_k$ $(k = 1, 2, \cdots, K)$, What is the loading limit per vehicle $Q_k (k = 1, 2, \cdots, K)$, Supply goods to customer $L$ points. The number of goods required for all customer points is $q_i (i = 1, 2, \cdots, L)$. $d_{ij}$ represents the distance from the demand point $i$ to the demand point $j$. $d_{ij} (j = 1, 2, \cdots, N)$ indicates the distance from the logistics centre to each customer point. Let $n_k$ represent the number $k$ of demand points delivered by the first vehicle, so $n_k = 0$ means that the $k$-th vehicle did not participate in the delivery, the demand point of the $k$-th vehicle distribution route is saved in set $R_k$, Demand point $r_{ik}$ represents the $i$-th demand customer in the driving path $k$, $r_{k0} = 0\text{on behalf of the logistics centre.}$ Because the first customer point on each delivery route is the logistics centre, so for $k = 1, 2, \cdots, K$, $r_{k0} = 0$ holds. According to different problems and situations, the setting of objective function is different. Most of them take the shortest route, the fastest speed and the least vehicles as the goal of solving the delivery route. The objective function of this paper is the shortest delivery route. The objective function formula is more concise.

$$\text{Min} F = \sum_{k=1}^{K} \left( \sum_{i=1}^{n_k} d_{r_{k(i-1)}r_{ki}} + d_{r_{kno}} \right) t(n_k)$$

$F$ is the sum of the distances of all vehicle delivery routes, and the objective function is to find the minimum value of $F$. Let $t(n_k)$ is a 0-1 variable, when the number of demand points on the $k$-th path is not 0. That is, when $n_k \neq 0$, the variable is 1. When there is no demand point on the $k$-th path, i.e. $n_k = 0$, this variable is set to 0. The accumulation formula realizes the accumulation of $k$ from 1 to $K$ to get the sum of the length of all delivery routes. The sum of the distances from the $i$-1 customer point to the $i$ $(i = 1, 2, \cdots, n_k)$ customer point delivered by the vehicle $k$ plus the distance from the $n_k$ customer point to the logistics center is shown in brackets. That is the total length of route $K$. Therefore, the 0-1 variable is used to ensure that when the vehicle participates in the delivery, the length of the delivery route is added to $F$. If you do not participate in the delivery, you can directly make the route length of the delivery 0 and do not add it to $F$. With 0-1 variable, the objective function formula is more concise.

3.1. Constraint Condition

(1) The sum of the demand of all customer points on each delivery route shall not be greater than the loading limit of vehicles on the route. $q_{r_{ki}}$ represents the demand of goods at the $i$-th customer point on the delivery route $k$. The accumulation formula can traverse all customer points $i$ delivered by vehicle $k$ from 1 to $n_k$. $Q_k$ is the maximum loading capacity of vehicles for route $k$.

$$s.t. \sum_{i=1}^{n_k} q_{r_{ki}} \leq Q_k$$

(2) The length of each delivery route shall not be greater than the limit of the route for the delivery vehicles. $k$ in formula (5) is each value between 1 and $K$. $D_k$ on the right side of the inequality represents the maximum route of the $k$th car. On the left side of the inequality is the sum of the distances from the $i$-1-th customer point to the $i$-th $(i = 1, 2, \cdots, n_k)$ customer point delivered by the $k$-th car plus the distance from the $n_k$ customer point to the logistics centre. The 0-1 variable $t(n_k)$ is also used here to simplify the formula. When the vehicle does not participate in the delivery, the limit of its driving length will not be considered.

$$\left( \sum_{i=1}^{n_k} d_{r_{k(i-1)}r_{ki}} + d_{r_{kno}} \right) \times t(n_k) \leq D_k$$

(5)
(3) The total number of customer points delivered by each vehicle is less than the total demand points. \( n_k \) represents the number of demand points delivered by the \( k \)-th vehicle, so that the value is in the range of 0 to \( L \).

\[ 0 \leq n_k \leq L \]

(4) A collection of all customer points on each path. The demand points on the \( k \)-th vehicle distribution route are stored in the set \( R_k \). When \( i = 1,2,\ldots,n_k \), \( r_{ki} \) represents the entire customer points on the \( k \)-th vehicle distribution route. Of course, we need to satisfy the customer point between 1 and \( L \).

\[ R_k = \{ r_{ki} \mid r_{ki} \in \{1,2,\ldots,L\}, i = 1,2,\ldots,n_k \} \]

(5) It is necessary that the requirements of each customer point are met. \( n_k \) is the number of customer points delivered by the first vehicle, so when \( k = 1,2,\ldots,K \), the sum of the total number of customer points delivered on all routes is solved. To make this value equal to \( L \).

\[ \sum_{k=1}^{K} n_k = L \]

(6) Each customer point can only be delivered by one vehicle. For different \( k \), there is no intersection in the corresponding customer point set \( R_k \).

\[ R_{k_1} \cap R_{k_2} = \emptyset, \forall k_1 \neq k_2 \]

3.2. Modeling
The mathematical model is shown in Table 1.

| Table 1. Mathematical model |
|-----------------------------|
| **MinF = \sum_{i=1}^{K} (\sum_{j=1}^{n_{ij}} d_{x_{ij}} + d_{y_{ij}}) \times t(n_k)** | Objective function: Solving the shortest distribution route |
| s.t. \( \sum_{j=1}^{n_{ij}} q_{ij} \leq Q_k \) | The sum of demand on each delivery route shall not be greater than the loading limit of vehicles on the route |
| \( \sum_{j=1}^{n_{ij}} d_{x_{ij}} + d_{y_{ij}} \times t(n_k) \leq D_k \) | The length of each delivery route shall not be greater than the limit of the route for the delivery vehicles |
| 0 \leq n_k \leq L | The total number of customer points delivered by each vehicle is less than the total demand points |
| \( R_k = \{ r_{ni} \mid r_{ni} \in \{1,2,\ldots,L\}, i = 1,2,\ldots,n_k \} \) | A collection of all customer points on each path |
| \( \sum_{k=1}^{K} n_k = L \) | The needs of each customer point are met |
| \( R_{k_1} \cap R_{k_2} = \emptyset, \forall k_1 \neq k_2 \) | Each customer point can only be delivered by one vehicle |

4. Application of Improved Particle Swarm Optimization in Vehicle Routing Problem

4.1. Algorithm Thinking
(1) Encoding particles. There are \( n \) demand points, and each demand point corresponds to a two-dimensional vector. In this space, particle \( i \) represents demand point \( i (i = 1,2,\ldots,n) \), its corresponding vector \( z_{ix} \) represents the vehicle number of distribution customer point \( i \), and \( z_{iy} \) represents the order of this customer point in the process of vehicle \( z_{ix} \) driving (customer point \( i \) is the \( z_{iy} \) demand point delivered by the vehicle with vehicle number \( z_{ix} \)).

(2) Decoding particles. Corresponding relationship between distribution vehicles and number: round the component \( z_{ix} \) of particle \( i \), then the vehicles \( j \) assigned by logistics centre for demand point \( ic \) can be obtained. The order of customer points passed by vehicle \( j \): using the circulation, find all the demand points \( i \) passed by distribution vehicle \( j \), sort and label the value of \( z_{iy} \) from small to large, and determine the order of customer points passed by vehicle \( j \).

(3) Selection fitness function. Fitness value is used to judge the performance of particles, and the value is determined by its function. For this algorithm, the fitness function can be expressed as formula (6).
\[ f(x) = \sum_{k=1}^{K} \left( \sum_{i=1}^{N_k} d_{r_{k(i-1)}r_{ki}} + d_{r_{kn_k}r_{kn}} \right) t(n_k) \]  

(6)

4. The termination condition of the algorithm. The end condition of the algorithm is set to meet the maximum number of searches.

4.2. Algorithm Steps

(1) Setting parameters. There are mainly learning factors \( c_1, c_2, w_{max}, w_{min} \), the number of individuals \( n \) and the highest number of iterations \( T_{max} \).

(2) Initial population. The random coding particle population initializes the order of the distribution vehicles and the customer points in the distribution route of each particle.

(3) Information decoding. The location information of each customer point is decoded to generate a path selection method, which can solve the fitness value of each customer point under multiple constraints.

(4) Update the optimal value. For each customer point that needs to be delivered, after updating its location information, it determines whether to update its own optimal experience value and group optimal experience value according to the fitness value.

(5) Update particle attributes. According to the improved formula, the parameters \( w', x \) and each individual \( v \) are updated.

(6) Cycle judgment. Determine whether the search times meet the maximum search \( T_{max} \). If yes, the optimal position of the particle is output. If no, return to step (3).

(7) Information decoding. Decoding is called for the location of each customer point, so the best dispatch scheme can be obtained.

5. Simulation Experiment

Analyze and code the path optimization selection of 1 centre, 2 vehicles (each vehicle's loading capacity limit is 15 tons) and 8 demand points. Distance between demand points and demand \( q_i (i = 1, 2, \ldots, 8) \). Fill in Table 2 (0 for logistics centre point, 1 to 8 for customer point)

| Demand point | distance/km | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--------------|-------------|---|---|---|---|---|---|---|---|---|
| 0            | 0           | 5.5 | 7.5 | 8.0 | 10.5 | 5.0 | 8.0 | 10.0 | 11.0 | 9.0 |
| 1            | 5.5         | 7.5 | 8.0 | 10.5 | 5.0 | 8.0 | 10.0 | 11.0 | 9.0 |
| 2            | 4.0         | 7.5 | 0   | 8.5 | 10.0 | 12.0 | 7.5 | 8.0 | 8.5 |
| 3            | 7.0         | 8.0 | 8.5 | 0   | 10.0 | 5.0 | 9.0 | 10.0 | 12.0 |
| 4            | 8.0         | 10.5| 10.0| 10.0| 0   | 10.0 | 7.5 | 7.5 | 10.0 |
| 5            | 13.0        | 5.0 | 12.0| 5.5 | 10.0| 0   | 8.0 | 10.0 | 7.5 |
| 6            | 11.0        | 8.0 | 7.5 | 9.0 | 7.5 | 8.0 | 0   | 7.0 | 12.0 |
| 7            | 15.0        | 10.0| 8.0 | 10.0| 7.5 | 10.0| 7.5 | 0   | 10.0 |
| 8            | 9.0         | 11.0| 8.5 | 11.0| 10.0| 7.5 | 12.0| 10.0| 0   |

| \( q_i \) | 2 | 1 | 2 | 2 | 2 | 1 | 4 | 1 |
|-----------|---|---|---|---|---|---|---|---|

PSO, improved method 1 and improved method 2 were used to calculate the individual. In order to compare the performance of each method more conveniently, the parameters are set as follows: particle population is 8, learning factor \( c_1 = 2.05, c_2 = 2.05 \), the initial value of inertia weight \( w \) is 1.5, \( r_1 \) and \( r_2 \) are random numbers between 0 and 1. For 1000 searches, the independent calculation is 10. The statistical results are shown in Table 3

| Algorithm                 | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
|--------------------------|----|----|----|----|----|----|----|----|----|----|
| Standard PSO             | 74 | 77.5 | 72 | 82.5 | 70 | 69 | 71 | 75.5 | 72 | 70 |
| Improved algorithm 1     | 70.5 | 73 | 71 | 69 | 71 | 70 | 68 | 68 | 72.5 | 67 |
| Improved algorithm 2     | 71 | 68.5 | 71 | 69 | 67 | 67.5 | 67 | 70.5 | 69 | 67 |

The results of Table 3 are summarized and table 4 is obtained.
**Table 4.** Relevant data of each algorithm running ten times

| Algorithm          | Average value | Maximum | Minimum value |
|--------------------|---------------|---------|---------------|
| Standard PSO       | 73.35         | 82.5    | 69            |
| Improved algorithm1| 70.0          | 73      | 67            |
| Improved algorithm2| 68.75         | 71      | 67            |

According to the results in Table 3 and table 4, when all parameters except inertia weight are the same, the results of the two improved algorithms in the example are better than those of the standard algorithm, and the convergence performance is better, and the linear decreasing inertia weight is better than the piecewise linear decreasing inertia weight. On the basis of the improved algorithm, it meets the requirements of balancing individual optimization performance.

6. Conclusion

A method to dynamically adjust the inertia weight according to the number of iterations is presented. Considering the individuality and colony of particles, the inertia weight is dynamically changed according to the iteration of the algorithm in the process of particle search, so as to improve the convergence speed of the algorithm, increase the diversity of the particle population, and avoid the premature phenomenon due to the strong individuality of the individual and falling into local optimum. This algorithm is used to solve the logistics distribution path optimization problem. Through the simulation experiment of the specific vehicle routing optimization problem, the results show that after the improvement of the algorithm, the optimization performance is enhanced, the optimization speed is accelerated, and its complexity is not increased, which greatly improves the performance of the original algorithm.

Acknowledgments

Authors wishing to acknowledge financial support of the start-up fund for doctoral research in Nanchang Institute of technology.

References

[1] Guo Kai and Lv Yang 2020 Optimizing Routing Path Selection Method Particle Swarm Optimization International Journal of Pattern Recognition and Artificial Intelligence 34(12)
[2] Si Han 2020 A new intelligent method for travel path recommendation based on improved particle swarm optimization International Journal of Computing Science and Mathematics 12(1)
[3] Xun Li and Dandan Wu et al 2020 An Improved Method of Particle Swarm Optimization for Path Planning of Mobile Robot Journal of Control Science and Engineering 2020
[4] Qiong Gao 2018 Intelligent Vehicle Global Path Planning Based on Improved Particle Swarm Optimization Open Access Library Journal 5(4)
[5] Zong-ran Dong and Yan Lin 2017 A particle swarm optimization based approach for ship pipe route design International Shipbuilding Progress 63(1-2) pp 59-84
[6] Puspitasari F H and Kurniawan V R B 2021 Designing Optimal Distribution Routes using a Vehicle Routing Problem (VRP) Model in a Logistics Service Provider IOP Conference Series: Materials Science and Engineering 1071(1)
[7] Yonghao Zeng 2019 Improved Particle Swarm Optimization Based on Tabu Search for VRP Journal of Applied Science and Engineering Innovation 6(2)
[8] Florian Arnold and Kenneth Sörensen 2019What makes a VRP solution good? The generation of problem-specific knowledge for heuristics Computers & Operations Research 106pp280-288
[9] Tao Shuaihan 2021 An Improved genetic algorithm for solving TSP Journal of Physics: Conference Series 1952(4)
[10] Sharma Sachin and Jain Vinod 2021 A Novel Approach for Solving TSP Problem Using Genetic Algorithm Problem IOP Conference Series: Materials Science and Engineering 1116(1)
[11] Hansknecht Christoph and Joormann Imke et al 2021 Dynamic Shortest Paths Methods for the Time-Dependent TSP Algorithms 14(1)