Solve the IRP Problem with an Improved PSO

Zelin Wang[1](✉), Shi Cheng[1], and Hu Peng[2]

[1] College of Computer Science and Technology, Nantong University, Nantong 226009, China
whwzl@whu.edu.cn

[2] School of Information Science and Technology, Jiujiang University, Jiujiang 332005, China

Abstract. It is difficult to solve the inventory-routing problem, because it is a NP hard problem. To find the optimal solution with polynomial time is very difficult. Many scholars have studied it for many years to find a good solving method. This paper analyzed the inventory-routing optimization problem. Then considered PSO has a good performance in solving combinatorial optimization problems. The PSO was improved to make it be suitable for solving discrete combination optimization problems. In order to improve the performance of the PSO algorithm to solve the inventory routing problem, this paper put forward dynamic adjustment of inertia weight and accelerator factor of the PSO, and introduced mutation operator in PSO. It is proved by numerical experiments that the proposed algorithm has certain performance advantages, and it also proves that the improved algorithm can improve the performance of the algorithm.

Keywords: Particle swarm optimization algorithm · Inventory routing problem · Inertia weight · Accelerated factor

1 Introduction

Inventory routing problem (IRP) is to determine the inventory strategy and distribution strategy. The inventory strategy aims to determine the distribution object and distribution number of goods in every planning period, and distribution strategy is to determine the commodity distribution route. IRP seeks to minimize the sum of inventory costs and distribution costs. The IRP problem is the combination of inventory problem and distribution problem, which is to solve these two problems on one platform at the same time. Because these two problems are the opposite problem, in the pursuit of the minimum inventory cost, it will inevitably bring the maximum distribution cost; On the contrary, if the pursuit of distribution cost is minimized, it will inevitably bring the maximum inventory cost. But at the same time solve the two problems is a very tough job, both the problem itself is N-P difficult problem. Especially when the customer number of distribution goods is more, and the customer's demand is stochastic demand conditions, the problem of IRP optimal strategy is often very complex. The solution of problem often makes delivery number, distribution interval and the distribution route lack of stability. This paper tries to +adopt the greedy algorithm and the improved discrete differential evolution algorithm to search IRP approximate optimal solution.

In the existing available literature, the literature [1] through theoretical analysis and proof, there is a 98.5% chance of getting the best of the problem with the strategy of fixed partition to solve the problem of inventory cost and distribution cost. The partition can effectively simplify the problem and reduce the difficulty of the problem. Therefore, this paper also adopts the idea of fixed partition, which does not need to cost a lot of cost for the low probability event. Literature [2] is the earliest IRP partition thought introduction. The author put the individual customer requirements decomposition, and allow multiple vehicles to serve a customer, so that in the actual operation of the problem there will be more difficult. Literature [3] improved the literature [2]. Each customer can only allow a vehicle to server in a delivery period, can't separate distribution, but the strategy is to seek the optimal solution, so that the scale of problem is relatively small size. Literature [4] adopts the classic Lagrangian relaxation algorithm to solve the IRP problem, which is complicated and not easy to implement. Moreover, with the increase of the size of the problem, the complexity of the algorithm increases exponentially.

In solving the questions of IRP, because of the complexity of the problem itself, when the problem scale is larger, to find the optimal solution is a very tricky question. Literature [5] to try using the variable neighborhood search heuristic intelligent methods to solve the problem of IRP. The IRP problem is solved in two phases. First using variable neighborhood search heuristic algorithm to solve the vehicle routing problem with limited capacity, and this stage is not considering the inventory cost, which purpose is to achieve a feasible initial solution. Then in the second stage the initial solution is optimized with iteration method, and the ideal results have been achieved. In literature [6], the author proposed a tabu search heuristic intelligent algorithm to solve the problem of the shortest path of the inventory, and compared the running effect of the algorithm with the effect of the original Lagrangian relaxation algorithm, and proved that this method proposed is far superior effect.

From literature [7–9], from one side we can see that the PSO, in recent years, has made significant effect, because of its global convergence and robustness in solving massive combinatorial optimization problems. In this paper, the problem of IRP is solved by using the greedy method and the improved discrete PSO algorithm.

2 Problem Model

2.1 Problem Description

For the logistics mode based on the supplier management inventory, a distribution center corresponds to multiple n customers scattered in different geographical locations, with $n = \{0, 1, 2, 3 \ldots N\}$ to represent the collection of customers, where 0 represents the distribution center. Assuming that the requirements of n customers for the product in the distribution center are random, but the needs of the customers are relatively independent, and the customers demand distribution is the same. Each customer's needs cannot be broken up. Assuming that the inventory cost of the distribution center is not considered, and the product assumption of the distribution center will not be out of stock. Assuming that the number of vehicles is unlimited and the driving ability is unlimited, and have the same loading capacity, and the demand of each customer will not exceed the load capacity of the vehicle. If the customer's order quantity exceeds the demand, the corresponding inventory cost will be increased. If the order quantity is less than the quantity demanded, it will cause the loss of goods. The vehicle starts from the distribution center and ends up at the distribution center. The goal of solving the problem is to seek inventory strategy and distribution strategy of the minimization sum of inventory cost and distribution cost.

2.2 Variable Definition and Problem Modeling

See Table 1.

Table 1. Variable definition

Variable	Implication
n	Customer number
ui	Random quantity demand of customer i
xi	Inventory of customer i
v	Vehicle number
C_v	Load capacity
c_{ij}	Delivery cost between the customer i and customer j
h_i	Unit inventory rate of customer i
P_i	Unit loss rate of customer i
d_i	Delivery quantity of customer i
c_i	Maximum inventory of customer i
$f_i(u_i)$	Density function of random demand of customer i
x_{ijv}	$x_{ijv} = \begin{cases} 1 & \text{delivery vehicle } v \text{ drive from customer } i \text{ to customer } j \\ 0 & \text{otherwise} \end{cases}$

Inventory cost:

$$H_{\mathrm{i}} = h_i \int_0^{x_i + d_i} (x_i + d_i - u_i) f_i(u_i) du_i + p_i \int_{x_i + d_i}^{\infty} (u_i - (x_i + d_i)) f_i(u_i) du_i \tag{1}$$

Delivery cost:

$$c_t = \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{v=1}^{v} c_{ij} x_{ijv} \tag{2}$$

IRP total cost model:

$$\min\ c_{all} = c_t + \sum_{i=1}^{n} H_i \tag{3}$$

s.t.

$$\sum_{v=1}^{v}\sum_{j=1}^{n} x_{ijv} = 1 \quad \forall i \in N \tag{4}$$

$$\sum_{v=1}^{\mathrm{V}}\sum_{i=1}^{n} x_{ijv} = 1 \quad \forall j \in N \tag{5}$$

$$\sum_{i=1}^{n} x_{i0v} = 1 \quad v = 1, 2, \cdots v \tag{6}$$

$$\sum_{i=1}^{n} x_{0iv} = 1 \quad v = 1, 2, \cdots \mathrm{v} \tag{7}$$

3 PSO Algorithm and Improvisation

3.1 PSO Algorithm Basic Idea

Assuming that the problem to be solved is a minimization problem, and the mathematical model of the problem is $min\ f\ (x_1, x_2, \ldots, x_n)$, where $x_j \in [L_j, U_j]$, and $1 \leq j \leq$ n. The $X(0)$ is the initial population. Let $X_i(t) = (x_{i1}(t), x_{i2}(t), \ldots, x_{in}(t))$ is the ith individual in the t-generation population, and the population individual is n-dimensional spatial structure, and the population size is *NP*. The particle swarm optimization algorithm (PSO) [7], which is a kind of superior performance of intelligent algorithms, was put forward by the Eberhart and Kennedy in 1995. The algorithm requires cooperation between individual in the population, using the outstanding individuals in the population as well as the best individual in their own history for the evolution of individuals, so as to achieve the population individual information sharing, and excellent individual competitive learning each other, to realize swarm intelligence, and guidance optimization of population evolution in the process of the whole [8, 9]. PSO algorithm using speed position search model, form is as follows:

$$\begin{aligned} V_{id}(N+1) = {} & W \times V_{id}(N) + c_1 \times rand() \times (P_{id} - V_{id}(N)) \\ & + c_2 \times rand() \times (P_{gd} - X_{id}(N)) \end{aligned} \tag{8}$$

$$X_{id}(N+1) = X_{id}(N) + V_{id}(N+1) \tag{9}$$

Equations (8) and (9) indicates that the position of the ith particle of n particles in the solution space of D dimension is $X_i = (X_{i1}, X_{i2}, \ldots, X_{id})$, the velocity is $V_i = (V_{i1}, V_{i2}, \ldots, V_{id})$, where $i = (1, 2, \ldots, n)$, and $d = (1, 2, \ldots, D)$. The fitness value is calculated by substituting *Xi* into the optimization objective function, and the optimal individual of the ith particle is $P_i = (P_{i1}, P_{i2}, \ldots, P_{id})$, called P_{best}, and the optimal individual of the whole particle swarm is $Pg = (P_{g1}, P_{g2}, \ldots, P_{gd})$, called Pgest. The particle swarm searches the entire solution space through velocity and position update. In [11], the *W* is the inertial weight, the c_1 and c_2 are acceleration factors, also known as individual and social learning factors, and a random number which is between 0 and 1 can gained by the rand() function, and *N* is the number of current iterations. The calculation ideas of individual optimal individuals and global optimal individuals are as follows:

$$P_{best}(\mathrm{t}+1) = \begin{cases} P_{best}(\mathrm{t}) \; if\, f\,(P_{best}(\mathrm{t})) < f(x_i(t+1)) \\ x_i(t+1), \; if\, f(x_i(t+1)) < = f(P_{best}(t)) \end{cases} \tag{10}$$

$$P_{gest}(t+1) = min\{P_1(t+1), P_2(t+1), \cdots, P_N(t+1)\} \tag{11}$$

Where, $f\,(.)$ is the objective function of the optimization problem, corresponding to the fitness function of PSO algorithm. The fitness function of different optimization problems is also different. When the algorithm updates the extreme, it is the priority to update the individual optimal value of the particles, and then to update the global optimal value according to the individual optimal value of all particles. Since the *Pgest* in the entire particle swarm is taken as the optimal position. The PSO described above is also called global PSO. If the optimal location found in the fixed neighborhood of each particle is *Pgest*, this is the local version of PSO.

The PSO algorithm is simple and requires fewer parameters to be controlled. The specific algorithm is as follows:

Step 1: Initialize the algorithm and set the initial values of various basic parameters (such as the number of particles, maximum iteration times, acceleration factor, inertia weight, etc.).
Step 2: Randomly generate particles of the initial position and velocity sequences;
Step 3: Use the objective function to evaluate the particle, and calculate the *Pbest* and *pgest* initial value;
Step 4: Use formulation (8) to update the particle's velocity;
Step 5: Use formulation (9) to update the particle's position;
Step 6: Calculate the fitness value of particles, and use the formulation (10) to update the particle's individual *Pbest* optimal position;
Step 7: Use formulation (11), according to the updated all particle *Pbest* values, to update *Pgest*;
Step 8: Judge algorithm whether meets the termination conditions or not (reach the specified number of iterations or already get the optimal value or meet other conditions), if judgment result is to satisfy the condition, go to step 9, or return to step 4;
Step 9: Record and output pgest value and stop the algorithm.

3.2 Improvement of Particle Swarm Optimization Algorithm for IRP Problem

Discrete Particle Swarm Optimization Algorithm

PSO algorithm in solving large-scale, multidimensional problems has natural advantages which avoided the inefficiencies of exhaustive search, also avoids the random search without purpose. It has the direction, global search algorithm with a purpose. It had been proved by theory and practical application that it is an excellent algorithm in many intelligent algorithms for solving large-scale, multidimensional combinational optimal problems.

This paper is to solve the IRP problem by step method, and the first step is to operate partition and optimize partition.

The second step is to change the PSO algorithm into discrete PSO algorithm. First of all, the traditional PSO algorithm which can solving continuous optimization problems was improved, made him into can solve a discrete combination optimization problems.

Because each partition obtained by the first step optimization is an independent loop of Hamilton, and each individual solution is a Hamilton loop, therefore, need to use PSO to solve partition optimization. The classical PSO algorithm must be improved for the particle's position, speed, and operating in the following the corresponding improvement.

1. The position can be defined as a Hamilton circle with all nodes. Assuming that there are N nodes, and the arc between them exists. The position of the particle can be expressed as sequence $x = (n_1, n_2, \ldots, n_N, n_1)$.
2. The velocity can be defined as the exchange set of particle position, which can be described as an ordered list of permutation sequence, denoted as: $V = \{(i_k, j_k), i_k, j_k \epsilon \{1, 2, \ldots, N\}, k \epsilon \{1, 2, \ldots, m\}\}$, where m is the number of velocity transformation. In the exchange sequence, first, the first exchange subset is performed. Then, the second exchange is performed, and so on.
3. The addition operation Position and velocity
   The operation is that a group of permutation sequences were acted on the position of a particle, in turn. The operation result is a new position.
4. Subtraction of position and position
   The result of subtraction of the particle position and position is too a new permutation sequence. For example, if $x = (1, 3, 2, 5, 4, 1)$, $y = (1, 2, 4, 5, 3, 1)$, then $x + s_1 + s_2 = y$, $s_1 = (5, 2), s_2 = (3, 2), y - x = \{(5, 2), (3, 2)\}$.
5. The addition operation of the particle velocity and velocity
   The result of two permutation sequences combinations is a new permutation sequence. That is, a new velocity.
6. The multiplication of real numbers and particle velocity
   Assuming that c is a real number, which value is a random, in (0, 1), and the velocity is a permutation sequence. The essence of the multiplication is to intercept permutation sequence, where the intercept value is int(c * k).

Dynamic Adjustment of the PSO Parameters

Because the inertia weight W has a great influence on the global search ability and local search ability of the PSO algorithm, therefore, the specific setting method of its value is worth studying. In the process of the operation of the algorithm. We hope that the global convergence capability is getting weaker and weaker, while the local search capability is getting stronger and stronger, so that we can give consideration to the both convergence of the algorithm and avoiding premature phenomenon. Clearly setting a fixed value to it does not balance the two factors. According to the analysis of the problem, the inertia weight W should be in reverse relation with the number of evolution generation, that is, with the gradual advance of the evolution, it gradually decreases.

In addition, the choice method of the acceleration factor c_1, c_2 can adopt random choice or purposeful choice. Appropriate choice method of them will improve the convergence speed of algorithm and avoid be caught into the local extremum.

In view of the problem of IRP, this paper used the cauchy distribution to dynamic adjust the inertial weight W. Cauchy distribution flanks are widely distributed, which is suitable for research of IRP. The algorithm can expand in the evolution process of the evolution of the range, so as not to fall into local optimum. Dynamic adjustment is carried out according to the following formula.

$$F = \begin{cases} 1 - \left| \frac{f_{ave} - f}{f_{ave} - f_{best}} \right| & if \quad f_{ave} > f \\ Cauchy(-2, 0.4) & otherwise \end{cases} \tag{12}$$

Where f is the fitness value of Xa(t) individual, and f_{ave} is the average fitness value of the current population, and f_{best} is the best fitness value of the current population.

The probability density function of cauchy distribution is

$$f(x) = \frac{1}{\pi}\left[\frac{0.4}{(x-2)^2 + 0.4}\right] \tag{13}$$

Where the dynamic adjustment of the inertial weight W is related to the fitness value, adjust dynamically according to the fitness value.

For C_1, C_2 acceleration factor dynamic adjustment, this paper adjusted it based on evolution generation. In the initial stages of the evolution. Let the C_1 smaller values, so that keep the diversity of population. Along with the evolution of advancing step by step, slowly increase the value of the C_1 to speed up the convergence speed of the algorithm. The C_1 is adjusted according to the following formula.

$$c1 = \begin{cases} \frac{g}{100-g} + 1 & c1 < 3 \\ 3 & otherwise \end{cases} \tag{14}$$

Where, g is the value of the current generation variable. Let C_1 value dynamic adjust tin the [1, 3]. In addition, $C_2 = 4 - C_1$.

Mutation Operation

Relative to the evolutionary algorithm, the particle swarm optimization has not crossover and mutation operations. Although the method is simple, it inevitably weakens algorithm's ability to control the global search and local search. In order to further improve the performance of particle swarm optimization algorithm, this paper introduced mutation operator into PSO algorithm.

In previous studies, random mutation and gaussian mutation are the most frequently used for mutation operator. The random mutations can significantly increase the algorithm's global search ability, but ignore the current solution, thereby reducing the performance of the PSO, and Gauss algorithm can enhance the local search ability of algorithm, also may let algorithm falls into local optimum, cause premature phenomenon.

The ability of particle swarm optimization algorithm has a lot to do with inertia weight W. The W Settings, in the last section, used cauchy distribution to dynamically adjust, so that in solving practical problems the performance of the algorithm was greatly improved. In order to increase the global searching ability in the early stage of evolution, random mutation operator is introduced into PSO algorithm. In order to increase the ability of local search, gaussian mutation operator is introduced into PSO algorithm. In order not to allow the algorithm to search everywhere aimlessly at the initial stage, random operators can be introduced in probability, and gaussian variation can be introduced in probability to avoid the algorithm falling into the local optimum. Formula (9) is improved as follows:

$$X_{id}(N+1) = \begin{cases} X_{id}(N) + V_{id}(N+1) & rand[0,1] < 0.5 \\ rand[x_{min}, x_{max}] & otherwise \end{cases} \tag{15}$$

$$X_{id}(N+1) = \begin{cases} X_{id}(N) + V_{id}(N+1) & rand[0,1] < 0.5 \\ P_{gest}(1 + Gauss(\sigma)) & otherwise \end{cases} \tag{16}$$

In the early evolution, using formula (15) instead of formula (9), and in the late evolution, using formula (16) to replace the formula (9). In this way, the algorithm not only considers the particularity of the individual, avoiding aimless global search, but, to some extent, enhances the global search ability of the algorithm. It is considered to avoid falling into the local optimum and to some extent enhance the local search capability in the later stage of the algorithm.

Therefore, this paper improves the standard PSO algorithm from three aspects, so as to guarantee the diversity of individual population and the convergence of the algorithm.

4 PSO Idea of Solving Stochastic Demand IRP Problem

4.1 The Client is Partitioned by Greedy Algorithm

First of all, according to the vehicle capacity and client demand quantity and the client's location coordinates, used a greedy algorithm to partition the customers, consequently get partition set, and then optimized the partition set by discrete differential evolution, consequently get S collection of partition.

IRP is a very complex problem of N-P. This article first to establish the coordinate system of coordinates dot with distribution center, and then sorted customer with location of abscissa x value from small to large, and then sorted customer with ordinate Y value from small to large. So we have a set of $X((Xm, y_1), (x_2, y_2), \ldots, (x_n, y_n))$. According to the initial demand quantity μ of each customer, algorithm partitioned customer. Partition started scanning from customer (Xm, y_1) in collection X, with the limit Cv vehicle capacity. The set X divided into M subset, respectively, the $XM((Xm, y_1), (x_2, y_2), \ldots, (x_i, y_i), X_2(), \ldots, Xm()$. The sum of customers demand quantity in each subset is not more than Cv. Let's say that this M subset is K_1.

Adjusted partitions, Surrounded each customer in a loop according to coordinate in the Fig. 1. The last element in the set Xm was transferred to the first element of the X_1 collection, and then checked whether the sum of all customers demand quantity in X_1 collection is more than vehicle capacity Cv, such as more than, transferred the last element of the X_1 set to X_2 set, become the first element in the X_2 set. Continued checking where the sum of all customers demand quantity in X_1 collection exceed the vehicle capacity, such as more than, continued to transfer the current last element of the X_1 to become the first element in the X_2. So repeatedly, until the sum demand quantity in the set X_1 did not exceed vehicle capacity. And then I was going to adjust X_2 as the method of adjustment X_1, and then I was going to check X_3 again, and I was going to do it again and again until M sets were checked out. Let's say that the set was K_2.

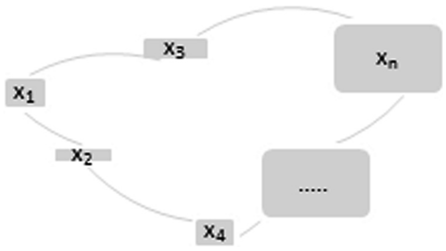

Fig. 1. Customer coordinate

Again to adjust set K_2 as the method of adjustment set K_1, and K_3 set was gotten, and so on, Ks sets were gotten. If the sum of demand of the last element in $X_{m\text{-}1}$ and the demand of all the elements of X_m did not more than vehicle capacity, put this element to the X_m set. Let's go to consider the penultimate element of $X_{m\text{-}1}$ until the sum of new elements demand which you consider and the all element demand of the new X_m set was more than vehicle capacity. No longer join, at this point, the number of elements in new X_m is S value.

4.2 The Improved Discrete PSO Algorithm is Used to Optimize the Partition

Through the first step processing, the *S* partition sets is obtained. The improved discrete difference evolution algorithm is adopted to optimize the path of s partition sets to find the optimal partition set of *S* partitions.

The algorithm step

```
The algorithm for looking for the optimal partition set
1  Initialization population number $NP$, generation number $g$
2  Initial position sequence and initial velocity sequence of population particles randomly or greedy
3  The particle was evaluated by using the objective function of the problem and the initial values of $P_{best}$ and $P_{gest}$ were calculated
4  According to formulation (12) (13) and (14), the $W$ $C_1$ $C_2$ were calculated
5  Formulation (8) is used to update the particle velocity
6  Formulation (14) or (15) is used to update the particle position
7  Calculation the fitness value of particle and update individual optimal position of particle
8  Update the $p_{gest}$ using the formulation (11),according to all the $P_{best}$
9  Judgment algorithm meets the termination conditions (reach the specified number of iterations or already get the optimal value or meet other conditions), if meets the termination conditions, algorithm reached the step 10, if not to return to step 4
10 Record and output $p_{gest}$ values and stop the algorithm
```

5 Numerical Experiment

5.1 Set Relevant Parameters

1. Set population scale NP = 50
2. Set evolution generation g = 100
3. Set customer number n = 10
4. C_{ij} is proportional to the distance between customer *i* and customer *j*, which is set the line spacing between them.
5. Set inventory rate $h = 1$, and the loss rate $p = 10$
6. Set vehicle delivery capacity $Cv = 600$
7. Set customer requirement $\mu < 200$, which follow a poisson distribution

5.2 Numerical Experiment

The Experiment Designed According to of Hybrid Algorithm Proposed in this Paper

According to the greedy algorithm of Sect. 3.1, and according to the coordinate value, client ranked as follows (Table 2):

(3, 9, 4, 7, 5, 10, 2, 6, 8, 1)
And then, according to the greedy algorithm, we got the first partition set K1:
((3, 9, 4, 7), (5, 10, 2)(6, 8, 1))
Finally, according to the greedy algorithm, a series of partition sets are obtained:
K2: ((1, 3, 9, 4), (7, 5, 10), (2, 6, 8))
K3: ((8, 1, 3), (9, 4, 7, 5), (10, 2, 6))

Table 2. Client dataSet

Client	Client coordinates	Client requirement	Maximum inventory
0*	(0, 0)	150	200
1	(85, 29)	199	200
2	(50, 60)	180	200
3	(5, 40)	177	200
4	(18, 5)	140	200
5	(30, 88)	166	200
6	(60, 49)	88	200
7	(25, 48)	90	200
8	(72, 30)	199	200
9	(10, 39)	160	200
10	(45, 76)	197	200

*Client 0 is delivery center, which inventory is set infinity.

Through the greedy algorithm of Sect. 3, we end up with three partition sets, *K1*, *K2*, *K3*. Furthermore, the discrete PSO algorithm of Sect. 3.2 is used to optimize the three partitions, to find the optimal distribution path, to find the optimal partition.

K1: ((3, 9, 7, 4), (5, 10, 2), (61, 8)); total cost: 125.41 + 207.042 + 200.39 = 532.842
K2: ((3, 9, 1, 4), (7, 10, 5), (2, 6, 8)); total cost: 210.831 + 200.532 + 193.307 = 602.79
K3: ((1, 8, 3), (9, 5, 7, 4), (10, 6, 2)); total cost: 210.852 + 195.652 + 196.392 = 602.896

Thus, the optimal partition set is evolved: *K1*, and delivery routing is: (0-3-9-7-4-0) (0-5-10-2-0) (0-2-6-8-0), total cost was 532.842.

The Experiment was Carried out With the Previous Ideas

According to [5], the results of the experiment were also obtained by the optimal set K_1, but the experiment time was 2.0375e−4s, and the experiment time of the paper algorithm was 1.99041E−4s.

According to [6], the results of the experiment were also obtained by the optimal set K_1, and the experiment time was 2.0535E−4s.

Experiment adopted the discrete PSO algorithm, which was improved by standard PSO algorithm. but the inertial weight W and accelerated factor C_1, C_2 were not adjusted by dynamic adjustment mechanism, but a series of fixed values are used for the experiment.

Table 3. Experimental results of different *W* adjustment methods

W adjustment method	Optimal partition set	Time consumption	Minimum cost
0.4	((1, 8, 3), (9, 5, 7, 4), (10, 6, 2))	2.0037E−4	604.82
0.5	((3, 9, 1, 4), (7, 10, 5), (2, 6, 8))	2.001E−4	602.93
0.8	((3, 9, 7, 4), (5, 10, 2), (61, 8))	3.407E−4	533.064
Self-adjustment	((3, 9, 7, 4), (5, 10, 2), (61, 8))	1.99041E−4	532.842

Table 4. Experimental results of different *C1* adjustment methods

C_1 factor adjustment method	Optimal partition set	Time consumption	Minimum cost
0.4	((3, 9, 7, 4), (5, 10, 2), (61, 8))	2.382E−4	533.072
0.6	((3, 9, 1, 4), (7, 10, 5), (2, 6, 8))	2.0108E−4	603.076
0.9	((1, 8, 3), (9, 5, 7, 4), (10, 6, 2))	2.0039E−4	605.059
Self-adjustment	((3, 9, 7, 4), (5, 10, 2), (61, 8))	1.99041E−4	532.842

Table 5. Experiment results of differential $X_a(t)$ setting strategy

$X_a(t)$ setting strategy	Optimal partition set	Time consumption	Minimum cost
Formulation (9)	((3, 9, 7, 4), (5, 10, 2), (61, 8))	2.3189E−4	532.842
Random mutation	((3, 9, 1, 4), (7, 10, 5), (2, 6, 8))	1.8311E−4	603.10
Gaussian mutation	((3, 9, 1, 4), (7, 10, 5), (2, 6, 8))	1.8294E−4	603.07
Strategy proposed by this paper	((3, 9, 7, 4), (5, 10, 2), (61, 8))	1.99041E−4	532.842

6 Analysis

Through the analysis of Sect. 4.1 experiment, it can be concluded that the algorithm proposed in this paper is practical feasible, and can find the optimal solution in many feasible solutions.

Through the analysis of Sect. 4.2 experiment, it shows that the proposed algorithm has the advantages of real, although use [5, 6] thinking also can obtain the optimal partition set, but time consuming compared with the algorithm of this paper, a little less. [5] time consuming is 2.0375E−4, but the experiment by the algorithm proposed by this paper is 1.99041E−4. Compared with the algorithm in this paper, which is 2.2% slower. [6] it takes 2.0535e−4s, which is 3.0 points slower than the algorithm in this paper.

Through the analysis of Table 3 experiments, this paper puts forward the adjustment method of the *W* weight. When *W* is set to 0.4 and 0.5, the algorithm will go into local optimum algorithm, but *W* is set to 0.8, although can obtain the optimal partition set, but takes up a lot of 1.349e−6, increased by 0.67% than the dynamic adjustment idea proposed by this paper.

Through the analysis of Table 4 obtained by experiment, this paper proposed the adjustment of the crossover of differential evolution method is effective. When C_1 is set to 0.6 and 0.9, algorithm search the local optimum. When C_1 is set to 0.4. The algorithm time consumption will take longer. Compared with the algorithm of this paper, it takes 19.6% slower.

Through the analysis of Table 5, the method proposed in this paper has some advantages. Compared with the random mutation, the method is relatively time consumption shorter. Compared with Gaussian mutation, it is relatively not easy to get into local optimization.

7 Conclusion

Through the anatomy of the IRP problem, it is known that it is a NP hard problem, and it is hard to find an ideal solution in a reasonable time. Considering the particle swarm optimization algorithm outstanding performance in solving large-scale, multidimensional performance on combinatorial optimization problems, this paper tried to adopt the particle swarm optimization algorithm to solve the problem of IRP. given the standard particle swarm optimization algorithm is to solve the problem of continuous, and the IRP problem is discrete problems, thus in this paper, the particle swarm optimization algorithm was improved to make it is suitable for solving discrete combination optimization problems.

Additionally according to the particularity of the IRP, in this paper, the particle swarm optimization algorithm is carried on the dynamic adjustment inertial weight *W* and acceleration factor, to make them with the gradually development of evolution of dynamic adjustment. The purpose is in the early stages of the evolution, the algorithm has more diversity, and along with the advancement of evolution, the convergence of the algorithm was gradually strengthen, such not only ensure the convergence of the algorithm, and also helps the algorithm falls into local optimum.

The feasibility of the proposed algorithm is verified by numerical experiments, and compared with the latest algorithm, the optimal partition set obtained by this algorithm is proved to be optimal. Finally, the experimental results show that the proposed method had the advantage of adjusting the inertial weight and the acceleration factor, and can balance the convergence and the diversity of the population.

Acknowledgments. The authors would like to thank the doctoral fund provided by NanTong university. This work is supported by the Natural science foundation of China (No. 61763019).

References

1. Anily, S., Bramel, J.: An asymptotic 98.5% effective lower bound on fixed partition policies for the inventory-routing problem. Discrete Appl. Math. **145**(1), 22–39 (2004)
2. Anily, S., Federgruen, A.: One warehouse multiple retailer systems with vehicle routing costs. Manag. Sci. **36**(1), 92–114 (1990)
3. Chan, L., Federgruen, A., Simchi-Levi, D.: Probabilistic analyses and practical algorithms for inventory-routing models. Oper. Res. Int. J. **46**(1), 96–106 (1998)
4. Rafie-Majd, Z., Pasandideh, S.H.R., Naderi, B.: Modelling and solving the integrated inventory-location-routing problem in a multi-period and multi-perishable product supply chain with uncertainty: lagrangian relaxation algorithm. Comput. Chem. Eng. **109**, 9–22 (2017)
5. Mjirda, A., Jarboui, B., Macedo, R., Hanafi, S., Mladenovic, N.: A two phase variable neighborhood search for the multi-product inventory routing problem. Comput. Oper. Res. **52**, 291–299 (2014)
6. Li, K.P., Chen, B., Sivakumar, A.I., Wu, Y.: An inventory routing problem with the objective of travel time minimization. Eur. J. Oper. Res. **236**, 936–945 (2014)
7. Tang, R.: Decentralizing and coevolving differential evolution for large-scale global optimization problems. Appl. Intell. **4**, 1–16 (2017)
8. Ghasemishabankareh, B., Li, X., Ozlen, M.: Cooperative coevolutionary differential evolution with improved augmented Lagrangian to solve constrained optimisation problems. Inf. Sci. **369**, 441–456 (2016)
9. Salman, A.A., Ahmad, I., Omran, M.G.H.: A metaheuristic algorithm to solve satellite broadcast scheduling problem. Inf. Sci. **322**, 72–91 (2015)
10. Wang, Z., Wu, Z., Zhang, B.: Packet matching algorithm based on improving differential evolution. Wuhan Univ. J. Nat. Sci. **17**(5), 447–453 (2012)