J Inf Process Syst, Vol.18, No.5, pp.665~676, October 2022
https://doi.org/10.3745/JIPS.01.0090

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Optimization Method of Knapsack Problem Based on BPSO-SA in Logistics Distribution

Yan Zhang*, Tengyu Wu, and Xiaoyue Ding

Abstract
In modern logistics, the effective use of the vehicle volume and loading capacity will reduce the logistic cost. Many heuristic algorithms can solve this knapsack problem, but lots of these algorithms have a drawback, that is, they often fall into locally optimal solutions. A fusion optimization method based on simulated annealing algorithm (SA) and binary particle swarm optimization algorithm (BPSO) is proposed in the paper. We establish a logistics knapsack model of the fusion optimization algorithm. Then, a new model of express logistics simulation system is used for comparing three algorithms. The experiment verifies the effectiveness of the algorithm proposed in this paper. The experimental results show that the use of BPSO-SA algorithm can improve the utilization rate and the load rate of logistics distribution vehicles. So, the number of vehicles used for distribution and the average driving distance will be reduced. The purposes of the logistics knapsack problem optimization are achieved.

Keywords
Binary Particle Swarm, Knapsack Problem, Logistics Distribution, Logistics Simulation, Simulated Annealing

1. Introduction

At present, heuristic algorithm can solve knapsack problem, it mainly includes genetic algorithm [1], ant colony algorithm [2], the greedy algorithm [3], the particle swarm optimization algorithm [4], and the simulated annealing algorithm (SA) [5]. In order to solve the binary knapsack problem, Haddar et al. [6] proposed an algorithm, which combined the quantum particle swarm optimization algorithm with the local search operators. In addition, Glover [3] considered the single constraint knapsack problem, so an improved greedy algorithm was introduced. Sachdeva and Goel [7] raised an improved genetic algorithm to settle 0/1 knapsack problem. Li et al. [8] uses a binary particle swarm optimization (BPSO) to solve the BKP. He et al. [9] proposed a novel bee colony method based on the full mapping function. However, the algorithms mentioned above have a common drawback, that is, they often fall into locally optimal solutions.

Being prone to premature convergence is the biggest drawback of basic BPSO. This paper proposes an algorithm to overcome the drawback of basic BPSO. SA has Metropolis criterion that makes the search process accept bad solution and jump out of local optimum, so this paper introduces SA operation in the process of particle swarm optimization.

Manuscript received February 15, 2022; first revision April 20, 2022; accepted May 23, 2022.
* **Corresponding Author:** Yan Zhang (16458321@qq.com)
Shool of Economics and Management, Chongqing, University of Posts and Telecommunications, Chongqing, China (16458321@qq.com, fly200205@163.com, s200301011@stu.cqupt.edu.cn)

Moreover, due to the dynamic update of the weight factor and the learning factor, the global and local search capabilities of the particles are effectively balanced in proposed integrated algorithm. And the experimental results show that, by comparing the other three algorithms, the integrated algorithm (BPSO-SA) has the best performance in terms of full load rate, total postage and total mass. So, the proposed algorithm can solve the knapsack problem better. It raises the load rate of vehicles and reduces the transportation costs, which are the main contributions of this paper.

The following four aspects including problem description, solution of knapsack problem in distribution, simulation results and analysis and conclusion will be discussed in the paper.

2. Problem Description

2.1 Description of Traditional Knapsack Problem

Knapsack problem was proposed by Merkel and Hellman [10] in 1978. The knapsack problem is a typical combinatorial optimization problem.

Description: M indicates the limited load of a knapsack, n represents the number of items, $W = [w_1, w_2, \dots, w_n]$ is the weight of the goods, and the corresponding value of the goods is $P = [p_1, p_2, \dots, p_n]$. Under the constraint condition of not exceeding the weight M that the knapsack can bear, find a solution that maximizes the total value of the object in the knapsack.

Mathematical model:

$$\begin{cases} Max \sum_{j=1}^{n} p_j \cdot x_j \\ s.t. \sum_{j=1}^{n} w_j \cdot x_j \leq M \\ x_j = 0 \; or \; x_j = 1 \end{cases} \tag{1}$$

where, $j = 1,2, \dots, n$, x_j represents that whether j item is placed in the knapsack, $x_j = 0$ represent j item is not placed into the knapsack. The constraint condition: $\sum_{j=1}^{n} w_j \cdot x_j \leq M$, the object function: $Max \sum_{j=1}^{n} p_j \cdot x_j$, p_j, w_j, M are all positive value.

2.2 Description of Knapsack Problem in Distribution

The knapsack problem is an important issue in the modern logistics distribution. The vehicles have the fixed volume and load. Given a set of orders, each with a weight and value, determine the number of items in a collection so that the total weight is less than or equal to the load of the vehicles and the total value is as large as possible. Each order has its own unique attributes in the process of distribution, including the quality wei, the volume vol and the grade gra of orders (the order is divided into two levels, the corresponding order postage val can be calculated according to the order level and quality), delivery vehicle has its fixed load limit M , rated capacity V and rated distance L . How to choose the combination of delivery orders to make full use of rated volume of the vehicle, realize the reasonable collocation of goods and reach the maximization of the total postage, load rate and volume rate in the case of not exceeding the weight limit of vehicle is the main task of the paper.

2.3 Particle Swarm Optimization Algorithm

The principle of particle swarm optimization algorithm is as follows: First, particle swarm searching in a D-dimension at a certain speed. Then, after every iteration, every particle recorded optimal position which denotes the local optimal value. The global optimal position indicates the best position of all the particles. $x_i = \left(x_{i,1}, x_{i,2}, \dots, x_{i,d}\right)$ represents the particle's position. $v_i = \left(v_{i,1}, v_{i,2}, \dots, v_{i,d}\right)$ represent the flight velocity of each particle. $pb_i = \left(pb_{i,1}, pb_{i,2}, \dots, pb_{i,d}\right)$ represent the current optimal of each particle. $gb_i = \left(gb_{i,1}, gb_{i,2}, \dots, gb_{i,d}\right)$ represent the best position of all particles. The process of updating each particle's flight speed and position is defined as the following formula.

$$v_{i,d}(t+1) = \omega \cdot v_{i,d}(t) + c1 \cdot r_1 \cdot \left(pb_{i,d} - x_{i,d}(t)\right) + c2 \cdot r_2 \cdot \left(gb_{i,d} - x_{i,d}(t)\right) \tag{2}$$

$$x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1) \tag{3}$$

where, t represents the number of iterations. ω represents inertia weight. The second term and the third term in Eq. (2) represent the particle's ability of local search and global search, respectively. c_1 and c_2 indicate acceleration coefficient. r_1 and r_2 denote random numbers between 0 and 1.

3. Solution of Knapsack Problem in Distribution

The SA algorithm can jump out of local optimum and the BPSO algorithm can search for the global optimum easily. A fusion optimization algorithm based on SA and BPSO is proposed in this paper to solve the knapsack problem in logistics distribution.

3.1 BPSO-SA

Being prone to premature convergence is the biggest drawback of basic BPSO. This paper proposes an algorithm to avoid the drawback of basic BPSO. SA has Metropolis criterion that makes the search process accept bad solution and jump out of local optimum. So, this paper introduces SA operation in the process of particle swarm optimization. Moreover, due to the introduction of the dynamic update of the weight factor and the learning factor, the global and local search capabilities of the particles are effectively balanced in proposed integrated algorithm.

3.1.1 Representation of solution

In the algorithm design process, the first step is to design solution representation scheme reasonably (a sequence is said an individual in the evolutionary process), standard 0–1 binary form is an obvious choice. So, this paper uses a 0–1 binary string with n bit to represent a solution of the algorithm as shown in Fig. 1.

d	*1*	*2*	*3*	*4*	…	*n-1*	*n*
pop[d]	0	1	0	1	…	0	1

Fig. 1. Representation of solution.

where, n is the number of variables, $pop_{i,d}$ = 1 said that the item d is packed into the knapsack.

3.1.2 Initialization of particle swarm

The algorithm uses the Eq. (4) to initialize the solution.

$$\begin{cases} pop_{i,d} = 0 & if\ rand < 0.5 \\ pop_{i,d} = 1 & else \end{cases} \tag{4}$$

3.1.3 Particle fitness update

The model has multiple objective functions. Because of the diverse dimensions of sub-objective functions, it is necessary to normalize the multi-objective functions. After transforming multi-objective into a single objective, the fitness value of the particle is updated. Vehicle load rate, vehicle volume rate, total postage and general objective functions are shown in Eq. (5).

$$\begin{cases} max\ Z_1 = \log\left(\frac{\sum_{j=1}^{n} w_j * x_j}{M}\right) \\ max\ Z_2 = \log\left(\frac{\sum_{j=1}^{n} v_j * x_j}{V}\right) \\ max\ Z_3 = \log\left(\sum_{j=1}^{n} val_j * x_j\right) \\ max\ f(x) = Z_1 + Z_2 + Z_3 \end{cases} \tag{5}$$

3.1.4 Particle search position and velocity updating

The model has multiple objective functions. Because of the diverse dimensions of sub-objective functions, it is necessary

$$\begin{aligned} \boldsymbol{v_{i,d}(t+1) = \omega * v_{i,d}(t) + c_1(t) * rand * \left(pb_{i,d} - pop_{i,d}(t)\right)} \\ \boldsymbol{+ c_2(t) * rand * (gb_{i,d}(t) - pop_{i,d}(t))} \end{aligned} \tag{6}$$

$$pop_{i,d}(t+1) = pop_{i,d}(t) + v_{i,d}(t+1) \tag{7}$$

$$\begin{cases} pop_{i,d}(t+1) = 1 & if\ rand < (pop_{i,d}(t+1) + v_{max})/(1 + 2 * v_{max}) \\ pop_{i,d}(t+1) = 0 & else \end{cases} \tag{8}$$

3.1.5 Adjustment of infeasible solution

Definition 1. In particle swarm evolutionary process, infeasible solutions generally refer to the candidate solutions which does not meet the knapsack constraints. This paper adjusts the infeasible solution to feasible solutions in order to meet the constraints. Specific adjustment method is as follows. Ranging the selected items based on the value sorting rate, and taking out the items at the end of the queue from the backpack until it meets the constraints of the backpack.

In this paper, the value rate of particles is calculated according to Eq. (9).

$$\delta_{i,j} = dat_{i,j} * \frac{wei_{i,j}}{vol_{i,j}} + val_{i,j} * gra_{i,j} \tag{9}$$

where dat is the date of the order, wei is the order of quality, vol is the order volume, val is the order postage, gra is the order level.

3.1.6 Update of weight and learning factors

In the search process, a larger weight ω has the advantage of jumping out of the local extreme points, and a smaller ω is good for fast convergence and high precision of search range. c_1, c_2 represent the learning factors, they adjust the local and global search ability of particle separately. BPSO-SA updates weight and learning factors according Eq. (10).

$$\begin{cases} \omega = \omega_{max} - (t * (\omega_{max} - \omega_{min})/gen_{max}) \\ c1 = c1_{max} - (t * (c1_{max} - c1_{min})/gen_{max}) \\ c2 = c2_{max} - (t * (c2_{max} - c2_{min})/gen_{max}) \end{cases} \tag{10}$$

where t denotes current evolutionary algebra, gen_{max} represents total evolutionary time.

3.1.7 Update of individual optimal solution and global optimal solution

The process of obtaining current particle's fitness value fit_i is described in this part. If fit_i is bigger than the current local or global optimal solution, then updating the local or global optimal solution to fit_i. If fit_i is smaller than the historical global optimal value, then generate a solution randomly. Then, calculating fit_i of the new solution. Using Metropolis criterion of SA to accept the solution, and update the global optimal solution correspondingly.

3.2 Algorithm Overall Process

3.2.1 Path adjustment

In [11], it takes the travel distance of vehicles and customer satisfaction as the objective functions. Kruskal algorithm is introduced to improve the basic genetic algorithm and the Kruskal Crossover Genetic Algorithm is proposed for logistics distribution path optimization. At the same time, the virtual logistics simulation system is built for testing the effectiveness of the algorithm optimization. This paper finds that many vehicles are far from reaching the fully loaded and the use of the vehicles capacity is not effective on the distribution paths which formed by above mentioned paper.

Pseudo-code of path adjustment implementation (Algorithm 1):

Algorithm 1. Path adjustment

```
Input: original path information
Output: New path
    Begin
    1: Enter the initialization data, $M$ (load limiting) $V$ (volume limit), $L$ (distance limit), distribution
       information $Distribution\_table$ of distribution center, the original path information matrix $Path$
    2: Sum the orders of each path and sort all paths according the total order quantity.
      For(Loop $N/2$ times)
          While $i < N$
              If $sum(Path, i) < M$ && $sum(Path, i+1) < M$
                  Combined $i$ and $i$+1 path to a path;
              End if
              $i=i+2$;
          End while
      End for
      Update path number $N$;
    End
```

3.2.2 Algorithm steps

The flow chart of the algorithm is shown in Fig. 2. The steps of the algorithm are as follows:

Step 1: Use the Eq. (4) to initialize the particle population (initial solution).

Step 2: Test whether the initial solution is satisfied with the constraint conditions, adjust it if there is infeasible solution.

Step 3: Use Eq. (8) to calculate the fitness value of particles (integrated objective function value), save the individual optimal solution, the global optimal solution.

Step 4: Use Eqs. (9)–(11) to update the position and velocity of particles simultaneously.

Step 5: Use Eq. (13) update the weight ω, $c1$, $c2$.

Step 6: Calculate the fitness value of the particle (objective function value), update the individual optimal solution and the global optimal solution.

Step 7: Update the global optimal solution by simulated annealing.

Step 8: If the iteration termination condition is reached, then output the optimal solution. Otherwise, jump to step 4, execute step 5–8 loopy.

Step 9: Output the optimal solution gb.

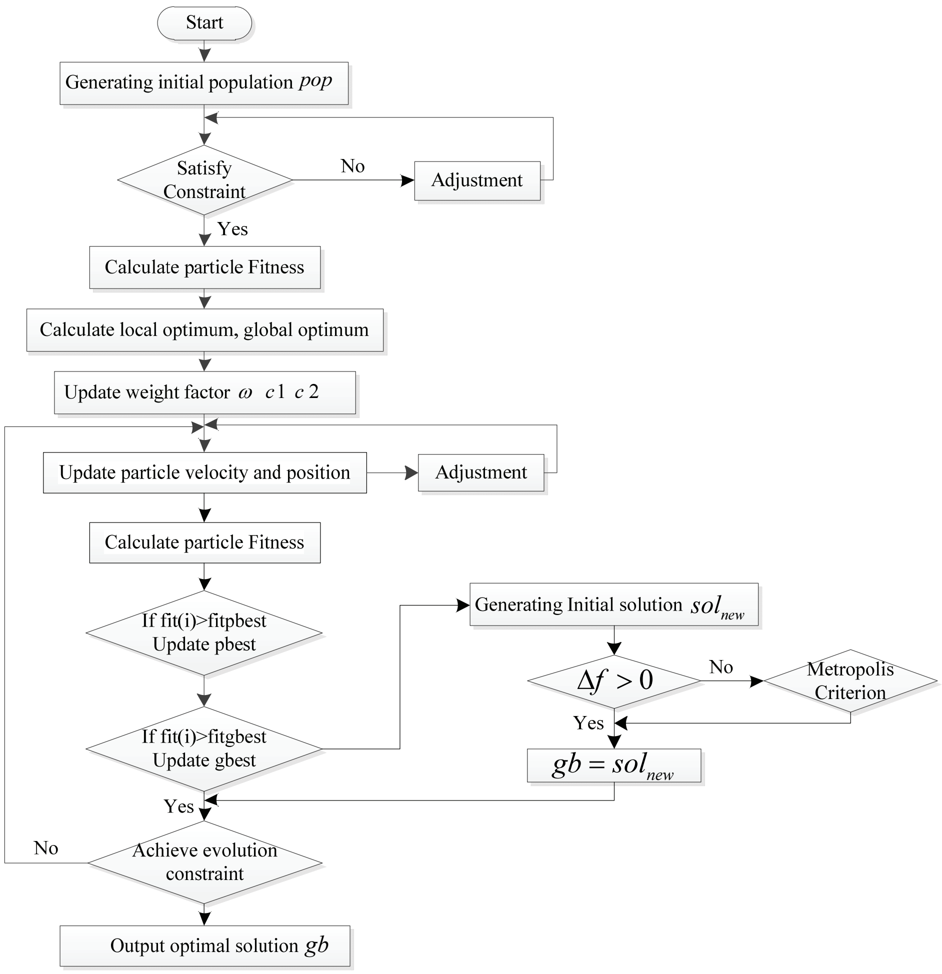

Start
Generating initial population pop
Satisfy Constraint
No
Adjustment
Yes
Calculate particle Fitness
Calculate local optimum, global optimum
Update weight factor ω c1 c2
Update particle velocity and position
Adjustment
Calculate particle Fitness
If fit(i)>fitpbest Update pbest
If fit(i)>fitgbest Update gbest
Generating Initial solution sol_new
Δf > 0
No
Metropolis Criterion
Yes
gb = sol_new
Yes
No
Achieve evolution constraint
Output optimal solution gb

Fig. 2. Algorithm flow chart.

3.3 Logistics Simulation System

In order to verify the effectiveness of the proposed algorithm, the intrinsic characteristics of logistics distribution operation are extracted and a virtual logistics simulation system is built in the paper. The virtual logistics simulation system includes the following seven main functional modules. They are system parameter configuration module, multi-tier city architecture generation module, order generation module, warehousing module, transportation module, distribution module and the most important knapsack optimization module. The overall operation architecture of the virtual logistics simulation system is shown in Fig. 3.

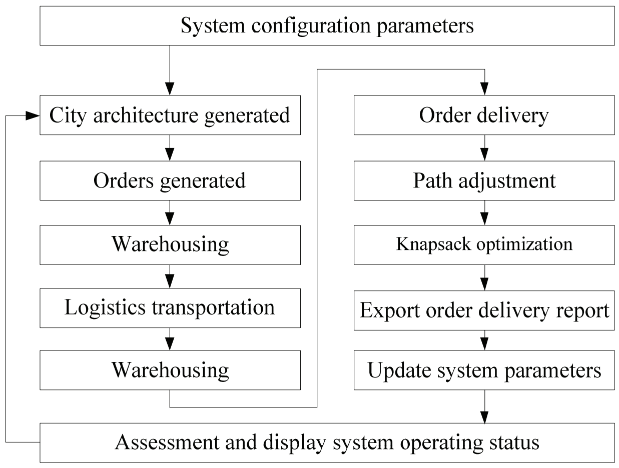


Fig. 3. The overall operation architecture of the virtual simulation system model.

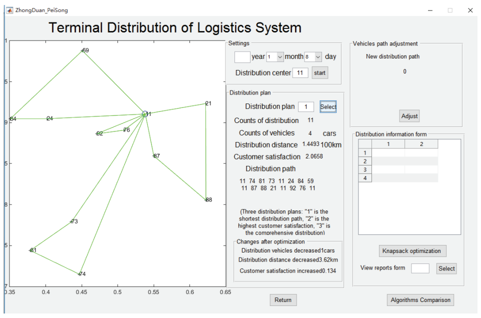
The operation principles of the simulation system are defined as follows. The system configuration parameters are set based on customer demands and city locations. The orders are gathered in the logistics centers (the first-level city points). Then the orders are transported to the distribution centers (the second-level city points). At last, the order arrive at the third-level city points where the customers are. We can see from Fig. 3, there are three steps to set the orders, which are system configuration parameters, city architecture generated and orders generated. When the orders are set, they will deliver from the logistics centers (the first-level city points) to the distribution centers (the second-level city points). During the order delivering from logistics centers to distribution centers, the orders are in the stages of warehousing, logistics transportation, warehousing and order delivery. The logistics centers warehouse the orders at first, then transport them to the distribution centers. When the distribution centers get the orders, they will warehouse the orders and then deliver them. When the orders are delivering from the distribution centers (the second-level city points) to the customers (the third-level city points), the orders are in the stages of path adjustment and knapsack optimization. After the process of path adjustment and knapsack optimization the order delivery report is carried out. After the orders get completely delivered, the system will update the parameters. The model is built on Windows 7 platform, using MATLAB2014 software.

4. Simulation Results and Analysis

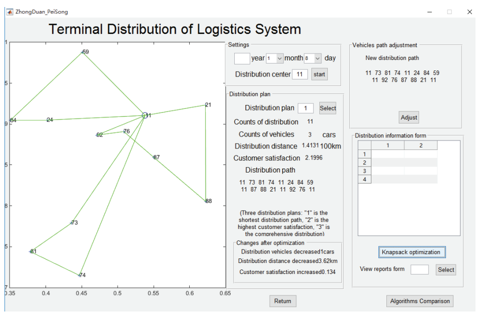
A distribution center and the distribution date are randomly selected. Firstly, the path that generated by the path optimization module is adjusted. Then, the comprehensive optimization algorithm BPSO-SA which proposed in this paper is used to optimize the loading of the orders in the distribution center. We compare the experimental results of distribution vehicles, distribution distance, average volume rate, average load rate and customer satisfaction, which get from before and after optimization.

(1) Simulation experiment of random distribution point

It can be seen from Fig. 4, in the test experiment, there are eleven distribution points which are randomly selected on January 8. Without using the BPSO-SA algorithm, to complete the distribution task, four vehicles are needed and the total distribution distance is almost 145 kM. The index of customer satisfaction is 2.066. After the optimization with BPSO-SA algorithm, only three vehicles are needed for delivering and the total distribution is 141 kM. The index of customer satisfaction is 2.200. The comparison results show that 1 distribution vehicle is cut down, 4 kM distribution distance is saved and the index of customer satisfaction is promoted by 0.13. After the optimization the number of distribution vehicle and the distribution distance is decreased, meanwhile the index of customer satisfaction is increased. The above results show that the algorithm can optimize the backpack problem well.



Counts of distribution 11
Counts of vehicles 4 cars
Distribution distance 1.4493 100km
Customer satisfaction 2.0658
Distribution path
11 74 81 73 11 24 84 59
11 87 88 21 11 92 76 11
(Three distribution plans: "1" is the shortest distribution path, "2" is the highest customer satisfaction, "3" is the comprehensive distribution)
Changes after optimization
Distribution vehicles decreased1cars
Distribution distance decreased3.62km
Customer satisfaction increased0.134



New distribution path
11 73 81 74 11 24 84 59
11 92 76 87 88 21 11
Counts of distribution 11
Counts of vehicles 3 cars
Distribution distance 1.4131 100km
Customer satisfaction 2.1996
Distribution path
11 73 81 74 11 24 84 59
11 87 88 21 11 92 76 11
(Three distribution plans: "1" is the shortest distribution path, "2" is the highest customer satisfaction, "3" is the comprehensive distribution)
Changes after optimization
Distribution vehicles decreased1cars
Distribution distance decreased3.62km
Customer satisfaction increased0.134

(a) (b)

Fig. 4. Comparison of the optimization results of the backpack: (a) before optimization and (b) after optimization.

(2) Simulation comparison experiment of different distribution points

We select the date and distribution centers randomly. The simulation data from before and after the backpack optimization in distribution centers are shown in Table 1.

The analysis data show that after the optimization, the number of distribution vehicles is reduced by an average of 2.3, the total distance decreased by an average of 7%, the average load rate increased by 18% and the average volume rate increased by 16%. But at the same time, the customer satisfaction was decreased by an average of 2%. The analysis data shows that the optimization algorithm reached the goal of improving the load of the vehicles.

(3) Simulation of the knapsack problem based on different algorithms in the simulation platform

In order to verify the effectiveness of the proposed algorithm, this paper compares the results of four algorithms (BPSO-SA, SA, BPSO. and GA) which are used to solve the same backpack problem in the built virtual logistics simulation platform.

As shown in Fig. 5, under the condition of the equal mass of orders, the vehicle loaded rate obtained by the BPSO-SA algorithm is 99.7%, SA is 98.7%, BPSO was 95.7% and GA was 89.2%. Thus, BPSO SA algorithm is obviously better than the other three algorithms.

Table 1. Different distribution center optimization results comparison table

Distribution point	Date	Distribution vehicle		Distribution distance (100 km)		Load rate (%)		Volume rate (%)		Customer satisfaction	
		Before	After	Before	After	Before	After	Before	After	Before	After
11	Jan 6	5.00	3.00	1.73	1.63	74	87	73	85	2.28	2.27
11	Jan 7	5.00	3.00	1.88	1.66	75	90	73	85	2.36	2.29
11	Jan 11	5.00	3.00	1.90	1.83	77	88	74	84	2.41	2.31
11	Jan 12	5.00	3.00	1.67	1.66	78	92	77	90	2.36	2.41
11	Jan 14	5.00	3.00	1.94	1.73	78	91	75	88	2.54	2.36
11	Jan 15	5.00	3.00	1.86	1.65	77	91	76	91	2.47	2.30
18	Jan 7	6.00	3.00	2.10	1.91	73	97	70	93	2.61	2.58
18	Jan 11	6.00	3.00	1.87	2.22	69	90	66	85	2.46	2.40
18	Jan 12	6.00	3.00	1.92	1.89	70	92	67	87	2.57	2.43
18	Jan 13	6.00	3.00	1.87	1.87	72	92	70	90	2.49	2.53
18	Jan 14	6.00	3.00	1.92	2.09	67	94	66	90	2.49	2.45
18	Jan 15	6.00	3.00	1.84	2.04	68	93	67	88	2.42	2.43
18	Jan 17	6.00	3.00	2.10	1.60	72	97	71	96	2.71	2.55
7	Jan 7	5.00	3.00	1.94	1.68	79	91	79	95	2.31	2.45
7	Jan 8	5.00	3.00	1.83	1.98	62	85	65	90	2.29	2.31
7	Jan 9	5.00	3.00	1.82	1.60	78	92	75	88	2.35	2.40
7	Jan 11	5.00	3.00	1.86	1.69	78	93	75	87	2.40	2.44
7	Jan 12	5.00	3.00	1.92	1.75	81	91	78	89	2.34	2.41
7	Jan 13	5.00	3.00	1.91	1.98	66	92	64	86	2.32	2.36
7	Jan 14	5.00	3.00	1.92	1.92	66	92	63	87	2.32	2.33
7	Jan15	5.00	3.00	1.94	1.85	79	87	79	79	2.33	2.29
Mean range		↓2.33		↓7%		↑18%		↑16%		↓2%	

The changing trend of the total message with postage algorithm iterative process is shown in Fig. 5. In the Fig. 5, the black point line presents the BPSO-SA algorithm postage objective function changing with the number of iterations situation. We can see from the chart, the BPSO-SA optimization algorithm convergence speed is faster than other algorithms and can obtain higher total postage.

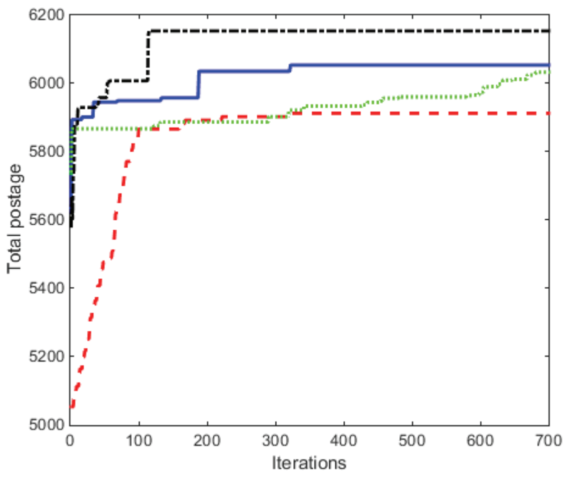


Fig. 5. Total change trend with postage algorithms.

The computational complexity of the BPSO-SA algorithm is also better than other three algorithms. It can be seen from Fig. 5, to reach the same total postage 6000, the BPSO-SA algorithm only needs 57 iterations, the SA algorithm needs 203 iterations, the BPSO algorithm needs 716 iterations and the GA algorithm cannot reach 6000. So, the BPSO-SA is more efficient than other three algorithms.

(4) Table 2 shows that the simulation results of the BPSO-SA algorithm are the best of the four algorithms (BPSO-SA, GA, BPSO, and SA) in terms of total mass, total postage, and full load rate under the same conditions. The average value of total mass fluctuates around 598.8, the maximum value is fixed at 600 and the minimum value is maintained at 594; the average value of total postage gradually decreases, the maximum value increases abruptly to 6199 and the minimum value decreases to 5980; the average value of full load rate gradually increases to 99.1%, the maximum value is fixed at 99.8% and the minimum value decreases to 97.0%.

Table 2. Simulation results of different algorithms under different running times

	Algorithm	Total mass			Total postage			Full-load ratio (%)		
		Avg	Max	Min	Avg	Max	Min	Avg	Max	Min
10	SA	597.60	600.00	587.00	5847.20	5930.00	5749.00	98.6	99.8	95.7
	BPSO	597.40	600.00	588.00	6084.40	6140.00	6008.00	98.6	99.5	97.3
	GA	593.80	600.00	581.00	5900.40	5998.00	5804.00	94.3	96.3	91.5
	BPSO-SA	598.80	600.00	597.00	6085.40	6162.00	6010.00	99.1	99.8	97.8
20	SA	596.15	600.00	584.00	5858.70	5943.00	5745.00	98.9	99.8	95.7
	BPSO	597.70	600.00	588.00	6072.90	6175.00	5989.00	98.3	99.8	96.2
	GA	594.35	600.00	581.00	5912.15	6033.00	5799.00	95.1	98.3	91.5
	BPSO-SA	598.70	600.00	595.00	6076.30	6162.00	5980.00	99.1	99.8	97.0
40	SA	596.08	600.00	584.00	5863.00	5993.00	5624.00	98.9	99.8	94.5
	BPSO	597.45	600.00	588.00	6068.60	6175.00	5977.00	98.5	99.8	95.3
	GA	594.13	600.00	570.00	5924.10	6070.00	5724.00	94.8	99.5	88.3
	BPSO-SA	598.78	600.00	594.00	6073.40	6162.00	5980.00	99.2	99.8	97.0
60	SA	596.10	600.00	575.00	5847.97	6011.00	5624.00	98.9	99.8	94.5
	BPSO	597.82	600.00	588.00	6068.46	6175.00	5977.00	98.5	99.8	95.3
	GA	593.64	600.00	570.00	5933.75	6128.00	5724.00	94.6	99.8	88.3
	BPSO-SA	598.84	600.00	594.00	6073.00	6199.00	5980.00	99.3	99.8	97.0

5. Conclusion

The existing algorithms for solving the logistics knapsack problem are easy to fall into local optimum. The BPSO algorithm has a good global search ability. The SA algorithm has an excellent local search ability. The proposed integrated optimization algorithm that combined the above 2 algorithms can avoid local optimum. The experimental results show that the proposed algorithm can effectively optimize the backpack problem by reducing the number of distribution vehicles and distance. The experimental results also show that, by comparing the four algorithms (BPSO_SA, GA, BPSO, and SA), the integrated algorithm (BPSO_SA) has the best performance in terms of full load rate, total postage and total mass.

Due to the limitations of the experimental conditions, the research in this paper is mainly carried out on the basis of the data generated by the simulation model. In the next step of research, the actual logistics data will be used for experiments to verify the validity of the proposed algorithm.

Acknowledgement

This paper is supported by the Science and Technology Research Program of Chongqing Municipal Education Commission (No. KJQN201900625 and KJQN202000638).

References

[1] J. H. Drake, M. Hyde, K. Ibrahim, and E. Ozcan, “A genetic programming hyper-heuristic for the multidimensional knapsack problem,” *Kybernetes*, vol. 43, no. 9/10, pp. 1500-1511, 2014.
[2] Z. Beheshti, S. M. Shamsuddin, and S. Hasan, “Memetic binary particle swarm optimization for discrete optimization problems,” *Information Sciences*, vol. 299, pp. 58-84, 2015.
[3] F. Glover, “Advanced greedy algorithms and surrogate constraint methods for linear and quadratic knapsack and covering problems,” *European Journal of Operational Research*, vol. 230, no. 2, pp. 212-225, 2013.
[4] I. B. Mansour and I. Alaya, “Indicator based ant colony optimization for multi-objective knapsack problem,” *Procedia Computer Science*, vol. 60, pp. 448-457, 2015.
[5] S. C. Leung, D. Zhang, C. Zhou, and T. Wu, “A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem,” *Computers & Operations Research*, vol. 39, no. 1, pp. 64-73, 2012.
[6] B. Haddar, M. Khemakhem, H. Rhimi, and H. Chabchoub, “A quantum particle swarm optimization for the 0–1 generalized knapsack sharing problem,” *Natural Computing*, vol. 15, no. 1, pp. 153-164, 2016.
[7] C. Sachdeva and S. Goel, “An improved approach for solving 0/1 knapsack problem in polynomial time using genetic algorithms,” in *Proceedings of International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, Jaipur, India, 2014, pp. 1-4.
[8] R. Merkle and M. Hellman, “Hiding information and signatures in trapdoor knapsacks,” *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 525-530, 1978.
[9] Y. Li, Y. He, H. Li, X. Guo, and Z. Li, “A binary particle swarm optimization for solving the bounded knapsack problem,” in *Computational Intelligence and Intelligent Systems*. Singapore: Springer, 2018, pp. 50-60.
[10] Y. He, H. Xie, T. L. Wong, and X. Wang, “A novel binary artificial bee colony algorithm for the set-union knapsack problem,” *Future Generation Computer Systems*, vol. 78, pp. 77-86, 2018.
[11] Y. Zhang, X. Y. Wu, and O. K. Kwon, “Research on Kruskal crossover genetic algorithm for multi-objective logistics distribution path optimization,” *International Journal of Multimedia and Ubiquitous Engineering*, vol. 10, no. 8, pp. 367-378, 2015.

Yan Zhang **https://orcid.org/0000-0002-7135-0740**

She received her B.E. degree in international journalism from Sichuan International Studies University in 2004, M.S. and Ph.D. degree in logistics from Inha University, South Korea in 2007 and 2018, respectively. She is currently a lecturer in School of Management, Chongqing University of Posts and Telecommunications, Chongqing, China. Her research interests include logistics management and logistics system optimization.

Tengyu Wu **https://orcid.org/0000-0001-8169-2675**

He received his B.E. degree in engineering management from Xi’an Jiaotong University in 2006, M.S. in Xi’an University of Technology, Ph.D. degree in Xi’an Jiaotong University. He is currently a lecturer in School of Modern Post, Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include logistics management and routing optimization.

Xiaoyue Ding **https://orcid.org/0000-0001-6358-9594**

She received B.E. degree in Internet of Things Engineering from Chongqing University of Posts and Telecommunications in 2020. Since September 2020, she is with the School of Automation, Chongqing University of Posts and Telecommunications, Chongqing, China as a M.S. candidate. Her current research interests include internet of vehicle and video transmission.