Brainstorming-Based Large Scale Neighborhood Search for Vehicle Routing with Real Travel Time

Jia Liu[1], Nanqing Guo[1,2], and Bowen Xue[1(✉)]

[1] College of Management, Shenzhen University, Shenzhen 518060, China
isxuebowen@163.com

[2] Greater Bay Area International Institute for Innovation, Shenzhen University, Shenzhen 518060, China

Abstract. It is vital to improve the efficiency of urban distribution with severe traffic congestion problems. However, few studies on urban distribution consider the impact of road conditions, which leads to less instructive results to the real-world problems. This paper considers the real travel time of the vehicles to minimize the total distribution time. Furthermore, hybrid brain storm optimization and large neighborhood search algorithm (BSO-LNS) are designed to solve this problem. The experimental results show that the proposed BSO-LNS algorithm is superior to peer competitors. The optimized routes are easier to obtain the global optima, which is suitable for solving the urban distribution problem under such complex road conditions.

Keywords: Real-time road conditions · Urban distribution · Crawler · Brain storm optimization · Large neighborhood search

1 Introduction

As an essential part of the last-mile distribution in the logistics chain, urban distribution plays a significant role in the entire supply chain. With the increasing prominence of urban governance problems, the role of urban distribution has become more prominent. Urban distribution is an important part of the Vehicle Routing Problem (VRP). Since Solomon and Desrosiers [1] put forward the concept of the time window in VRP, there is increasing research on Vehicle Routing Problem With Time Windows (VRPTW). Furthermore, Jabali [2] proposed the concepts of the soft time window and hard time window, and introduced the penalty function into the objective function to solve the VRPTW, thus giving birth to considerable research on urban distribution [3–7]. At the present stage, with the acceleration of the urban process, the road traffic conditions are becoming more and more complex, which puts forward higher requirements for urban distribution. The traditional path planning represented by Euclidean distance can no longer meet the current distribution demand. At this stage, urban distribution needs to consider the impact of road conditions and develop a more efficient algorithm to solve the problem of urban distribution better in reality.

Y. Xu et al. (Eds.): ML4CS 2022, LNCS 13657, pp. 358–370, 2023.
https://doi.org/10.1007/978-3-031-20102-8_28

In recent years, there has been more research on real-time road conditions. Some scholars consider the dynamics of travel time through the system's design to receive real-time road network information in the distribution process, automatically adjust the distribution route and upload it to the distribution driver [8]. Thanks to the development of big data collection technology, obtaining historical road condition data is no longer challenging. We can collect real-time road condition information in the process of a vehicle driving through the electronic map platform [9]. By collecting road condition data in a specific time range, the designed algorithm is used to predict the driving speed of each road section [10] to realize the route change. As the problem becomes more and more complex, some scholars choose to design a more efficient algorithm [11] through reasonable planning of the departure time of each car to avoid congested roads [12] effectively.

In this paper, we use crawler technology to obtain the real travel time through Amap API and choose a hybrid algorithm (BSO-LNS) that combines brainstorming optimization algorithm (BSO) and large neighborhood search algorithm (LNS) to solve this kind of distribution optimization problem. The large neighborhood search algorithm is an excellent local search algorithm, which is proposed by Shaw [13] and is often used to solve VRPTW. Although the brainstorming algorithm is relatively young, it has received widespread attention since it was proposed by Shi [14] in 2011, and it has also been used to solve the VRPTW in recent years. However, few papers use BSO alone to solve VRPTW [15,16], and more often, they are hybrids with other algorithms [17–20], all achieving better results than BSO algorithm.

The next section of this paper is organized as follows. Section 2 focuses on the basic process of combining the BSO and the LNS algorithm, Sect. 3 develops a VRPTW model to find the shortest total time spent, Sect. 4 gives the comparison results of the experiments, and the last section is used to conclude and look ahead.

2 BSO-LNS for VRPTW

2.1 Problem Description

Based on real-time road conditions, the urban distribution problem can be described as follows: a department store distribution center located in an urban area needs to deliver goods to several supermarkets in the surrounding area, and the geographic location information of the distribution point can be obtained from Amap, the number of goods required to be delivered by each supermarket on that day is prepared by the distribution center at least one day in advance, and the vehicles loaded with their respective responsibilities for the distribution of supermarkets set off from the distribution center toward the planned route, when arriving before the left time window of supermarkets, they need to wait until the service start time to unload the goods, and if they exceed the right time window, they will be penalized in the target function. At the same time, under the influence of considering the actual road conditions, the constant speed

of each vehicle and the distance between the two is no longer assumed here to be counted as the Euclidean distance, still, the passage time and distance between the two are obtained in real-time through Amap, the corresponding passage time and distance matrix is obtained, aiming for the optimized delivery solution with the least total driving time to be closer to the actual delivery scenario under the urban congestion environment. In addition, the following assumptions are made for the problem to be solved.

(1) Each vehicle serves only one route, each node can only be served once by one vehicle, the vehicle type is the same and the load capacity is known, the total number of vehicles, the maximum mileage is not limited.
(2) The vehicle departs from the distribution center and finally returns to the distribution center within the time window.

2.2 Model Building

The VRPTW model considering the road conditions is constructed as follows:

$$\min Z = \sum_{k=1}^{K}\sum_{i=0}^{n}\sum_{j=0}^{n} x_{ijk}t_{ij} + \alpha \cdot \sum_{k=1}^{K} \max[(\sum_{i=0}^{n}\sum_{j=0}^{n} x_{ijk} - G), 0] + \beta \cdot \sum_{i=1}^{n} \max[(l_{ik} - b_i), 0] \tag{1}$$

$$\sum_{k=1}^{K}\sum_{i=0}^{n} x_{ijk} = 1; \forall j \in N \tag{2}$$

$$\sum_{i=0}^{n} x_{0ik} = 1; \forall \mathrm{k} \in \mathrm{K} \tag{3}$$

$$\sum_{i=0}^{n} x_{ijk} - \sum_{i=0}^{n} x_{ijk} = 0; \forall j \in n, k \in K \tag{4}$$

$$\sum_{i=0}^{n} x_{i0k} = 1; \forall \mathrm{k} \in \mathrm{K} \tag{5}$$

$$a_0 \le u_{0k} \le b_0; \forall k \in K \tag{6}$$

$$x_{ijk} = 0 or 1 i, j = 0, 1, ...n; \forall k \in K \tag{7}$$

The variables used in the model are shown in Table 1.

In the model, (1) is the objective function, which indicates the minimum total vehicle travel time and penalty cost for late arrival and overloading [17]; the constraint (2) indicates that each customer point can only be distributed by one vehicle; constraint (3) (4) (5) indicates that the input and output of distribution vehicles must satisfy the flow balance constraints; constraint (6) indicates that each vehicle starts from the distribution center and returns to the distribution center within the time window of the distribution center after completing the distribution; constraint (7) denotes the corresponding 0–1 constraint.

Table 1. Variable description.

Symbols	Description
i, j	$i, j = 0, 1, ..., n$. 0 means the distribution center,1∼n denotes supermarkets
k	Vehicle no. $k = 1,2,\ldots,K$
G	Maximum loading capacity of vehicle
t_{ij}	The actual passage time of vehicles between nodes, obtained by the crawler
a_i	The earliest service time of supermarket i
b_i	The latest closing time of supermarket i
s_i	Receiving service time of supermarket i
a_0	The earliest departure time of a vehicle at a distribution center
b_0	The latest time for the vehicle to return to the distribution center
g_i	The demands of supermarket i
M	denotes a large enough positive number for the penalty constraint
x_{ijk}	If truck k from node i to node j, then $x_{ijk} = 1$, else $x_{ijk} = 0$
l_{ik}	The time of vehicle k arrives at supermarket i
u_{ik}	The time of supermarket i began to accept the service of vehicle k, which i = 1, 2, . . . , N

2.3 Coding Analysis

In this paper, integer coding is used for distribution schemes to code individuals. To indicate simplicity, if there are now nine supermarkets to be delivered and there are at most three vans available in the distribution center at present, the individual codable schemes can be roughly divided into three cases according to the number of vehicles used.

(1) If three vehicles are dispatched to perform distribution tasks at the same time, one possible way for an individual to behave is as Fig. 1. Where numbers 1–9 denote supermarkets to be delivered, 10 and 11 denote distribution centers.

Fig. 1. One of situation with three vehicles.

Then, from Fig. 1, it can be seen that distribution centers 10 and 11 divide all individuals into 3 distribution routes and the distribution scheme can be expressed as follows, where 0 denotes the distribution center.
Distribution route 1: 0 → 1 → 3 → 5 → 6 → 0
Distribution route 2: 0 → 2 → 7 → 9 → 0
Distribution route 3: 0 → 4 → 8 → 0

(2) If only two vehicles are assigned to the distribution task, one possible representation of the individual is as Fig. 2

Fig. 2. One of situation with two vehicles.

Then, as can be seen from the Fig. 2, the distribution scheme is expressed as follows.
Distribution route 1: $0 \rightarrow 1 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 0$
Distribution route 2: $0 \rightarrow 2 \rightarrow 7 \rightarrow 9 \rightarrow 8 \rightarrow 4 \rightarrow 0$

(3) If only 1 vehicle is assigned to the distribution task, one possible way for an individual to behave is as Fig. 3

Fig. 3. One of situation with one vehicle.

Then, as can be seen from the Fig. 3, the distribution scheme is as follows.
Distribution route: $0 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 1 \rightarrow 2 \rightarrow 7 \rightarrow 9 \rightarrow 8 \rightarrow 4 \rightarrow 0$

From the above coding method, it can be seen that if the number of supermarkets to be delivered is N and the distribution center can arrange at most K vehicles for delivery, then the individual coding can be expressed as a permutation of $N + K - 1$ numbers. Next, a population consisting of individuals according to the above encoding can be randomly generated. The fitness function chosen for this algorithm is the reciprocal of the objective function value.

3 Brain Storm Optimization with Large Neighborhood Search

3.1 Brain Storm Optimization

Inspired by the brainstorming session, an act of human brainstorming and bursting with creative inspiration, the brainstorming optimization algorithm was first proposed by shi [14] in 2011. The algorithm received much attention once it was proposed because it has the advantages of simple structure and few hyperparameters, which are ideal for solving high-dimensional multi-peak problems.

The basic idea of the BSO algorithm is to map each person's idea to each individual in the population and represent it as a feasible solution in the problem-solution set. The specific steps are as follows:

Step 1: Random initialization. Generate n individuals as the initial solution for the population. Step 2: Evaluation. The fitness function values are calculated

for these n individuals. Step 3: Clustering. The initially generated populations are clustered and grouped into m categories using the K-means method. And each individual in these m categories is ranked according to the fitness function value, and the best individual in each category is selected as the class center. Step 4: Generate new individuals. The generated new individuals follow the Eqs. (8) and (9) are updated, and the better individuals among them are saved. Step 5: Update the population. Determine whether the iteration termination condition is satisfied. If not, continue iterating and updating until the loop termination condition is satisfied.

$$X_{new}^{d} = X_{select}^{d} + \xi f(\mu, \sigma) \tag{8}$$

$$\xi = \log sig((0.5 \cdot \max\_iter - cur\_iter)/k) \cdot rand() \tag{9}$$

where $f(\mu, \sigma)$ is the normal distribution function, $\log sig$ is the logarithm of the sigmoid, and max_iter and cur_iter are the maximum number of iterations and the current number of iterations.

3.2 Large Neighborhood Search

Large Neighborhood Search(LNS) is an algorithm that uses two core ideas, "destroy" and "repair" [13]. The two core ideas of LNS are "destroy" and "repair", which simply means that the destroy operator destroys some individuals in the current solution, and then the repair operator is used to repair the destroyed solution. The advantage is that after destroying and repairing, we can get the full ordering of the initial solution and traverse the solution space of more problems.

The destroy steps are as follows:

Step 1: Firstly, we construct two sets, one is the set S for storing customers of distribution routes, and the other is the set V for temporarily storing removed customers. Step 2: First select a customer i from the set S at random and deposit it in the set V. Step 3: Choose any customer j from the set V, and then calculate the correlation between the remaining customers in the set S (denoted as S') and customer j are calculated separately, and then sorted according to the relevance from largest to smallest, and finally according to the formula $\text{int}(rand^{D} \cdot S')$ to decide the next customer to move out, the general value of D between 5 and 20, used to control the value is not exactly the way to maximize relevance. This process is repeated until the number of customers removed reaches a predetermined value.

The repair steps are as follows:

Step 1: Each insertable customer has an uncertain number of insertion positions, and the fitness difference is calculated for each possible case and the solution after being destroyed, and recorded. Step 2: Sort the fitness difference from largest to smallest, choose the insertion position with the largest difference to insert the customer back, and repeat the process until all the removed customers are inserted back.

3.3 BSO-LNS

The basic idea of combining BSO and LNS algorithm is that after BSO performs the individual update operation, it selects some of the worse individuals to "destroy" and "repair" according to the fitness so that the selected solutions are at least no worse than the original ones. Finally, the two parts of individuals are merged and iterated until the end of the cycle. More details are shown in Algorithm 1.

Algorithm 1: BSO-LNS Algorithm

```
Initialize parameters;
Initialize population;
while not terminated do
    evaluating individuals;
    k-means for population;
    update individuals;
    evaluating individuals;
    select poor individuals in top n% fitness;
    perform destory and repair for selected individuals;
    merge the updated individual with the original remaining individual;
    evaluating individuals;
    perform global individual update;
end
```

4 Experiments

4.1 Experimental Description

The data used in this experiment are provided by a distribution company, as shown in Table 2 and Table 3 (the contents not listed in the table are the same as Table 2.), which represent the order data of two days, respectively, where node 1 denotes the distribution center and nodes 2 to node 21 denote the supermarkets to be delivered, and the delivery vehicles are known to have a capacity of 7 tons, allowing a maximum of 10 vehicles for delivery. All experiments were conducted using MATLAB (R2018b) on an Intel(R) Core(TM) i7-8565U CPU @ 1.80 GHz with 8.00 GB of RAM.

4.2 Experimental Setup and Analysis of Results

In order to study the effect of road conditions on distribution, three experiments were carried out. In Experiment 1 and Experiment 2, the asymmetric passage time matrices of vehicles between nodes were obtained from AMap via Python program at 6:00 pm on 5.28 and 6.10, respectively, under the scenario of considering road conditions, and the acquisition was set to consider road conditions;

Table 2. Data set of Experiment 1.

Node	1	2	3	4	5	6	7
Longitude	115.971	115.9379	115.9028	115.838	115.9170	115.9053	115.8417
Latitude	28.6959	28.5315	28.6161	28.7578	28.6548	28.6310	28.6653
Demand /t	0	1.4269	0.3998	0.7220	3.2	1.1348	0.58
Service	0	21.40395	5.997345	10.8309	25	17.022	8.7
Left time	7:00	8:00	8:30	9:00	8:30	8:00	8:00
Right Time	21:00	16:00	17:00	16:00	11:00	17:00	17:00
Node	8	9	10	11	12	13	14
Longitude	115.807	115.9618	115.8104	115.9291	115.8936	115.9038	115.9107
Latitude	28.6752	28.6633	28.6959	28.5575	28.5852	28.6762	28.7476
Demand /t	0.22723	0.55351	1.528	3.4688	1.05399	0.49	3.28
Service	6.40854	8.30274	22.92	21.6262	15.80994	7.35	29.2
Left time	8:00	9:30	8:00	8:00	8:30	8:00	8:00
Right Time	17:00	16:00	17:00	16:00	17:00	16:00	16:00
Node	15	16	17	18	19	20	21
Longitude	115.948	115.8066	115.9469	116.1404	115.8710	115.8996	115.9192
Latitude	28.85	28.6923	28.6345	28.6558	28.6172	28.6739	28.6853
Demand /t	0.74466	2.16381	1.6	3.98634	3.21517	0.6	2.2077
Service	11.1699	32.45718	24	29.7951	28.22755	9	23.1167
Left time	8:00	8:00	8:00	8:00	8:00	10:00	8:00
Right Time	17:00	16:00	17:00	17:00	17:00	17:00	16:00

Table 3. Data set of Experiment 2 and 3.

Node	1	2	3	4	5	6	7
Demand /t	0	1.36	1.78	0.73	0.8	0.69	0.54
Service time	0	15	20	10	7	6	5
Node	8	9	10	11	12	13	14
Demand /t	2.38	0.56	2.59	1.56	0.87	1.49	0.36
Service time	24	5	25	16	8	15	5
Node	15	16	17	18	19	20	21
Demand /t	1.56	0.56	1.08	0.71	1.94	2.28	1.97
Service time	17	5	10	8	20	28	20

Experiment 3, on the other hand, did not consider road conditions, but used the traditional straight-line distance for calculation.

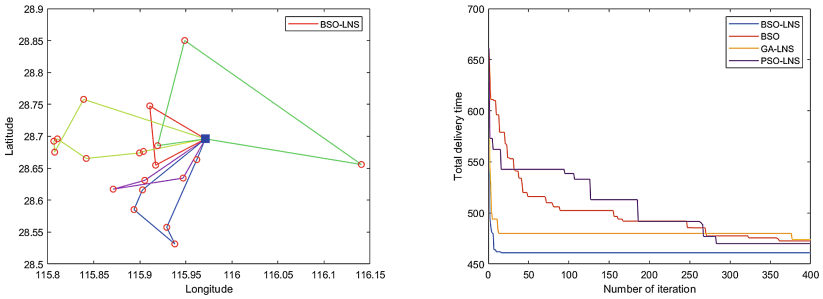
Meanwhile, to better reflect the performance of the BSO-LNS algorithm, the original BSO, GA-LNS and PSO-LNS algorithms are used for comparison in each experiment, and the common parameter settings for these experiments are shown in Table 4. The parameters in the BSO algorithm are set as follows: $m = 5, p1 = 0.1, p2 = 0.5, p3 = 0.5, p4 = 0.3, p5 = 0.2$. The parameters in the GA algorithm are set as follows: $pc = 0.9, pm = 0.05$. The parameters in the PSO algorithm are set as follows: $c1 = 1.5, c2 = 2.0, w = 1$.

Table 4. Common parameter setting for these algorithms.

Algorithm	N	Max iter	α	β	D
BSO-LNS	60	400	20	100	15
BSO	60	400	20	100	–
GA-LNS	60	400	20	100	15
PSO-LNS	60	400	20	100	15

Table 5. Simulation results of BSO-LNS with BSO, GA-LNS and PSO-LNS of 10 runs.

Data	Opt.	Alg.	Best	Worst	Avg	Var
Experiment1	461	BSO-LNS	461	461	461	0
		BSO	464.81	491.5	475.97	71.33
		GA-LNS	464	480	470.8	18.16
		PSO-LNS	463	505.87	487.04	150.85
Experiment2	578	BSO-LNS	578	581	578.7	1.21
		BSO	585	610.2	598.34	46.7
		GA-LNS	588	605	595.6	33.64
		PSO-LNS	583	610.6	595.38	49.88
Experiment3	3974.64	BSO-LNS	3974.64	3974.64	3974.64	0
		BSO	4038.59	4194.59	4112.41	2030.5
		GA-LNS	3974.64	3974.64	3974.64	0
		PSO-LNS	3995.52	4132.11	4050.15	1485.64

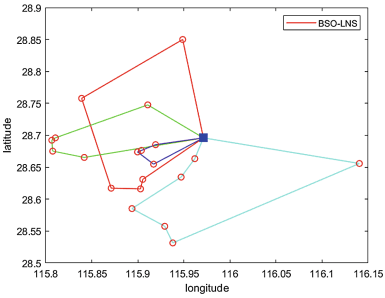


(a) BSO-LNS algorithm optimal solution

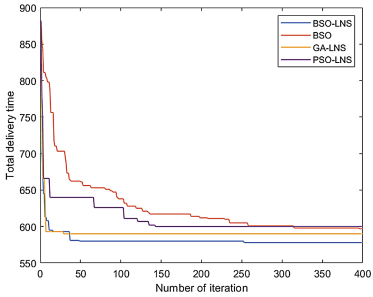
(b) Performance comparison

Fig. 4. Experiment 1.

We run the four algorithms each 10 times for each experiment, and show the results in Table 5. From Table 5, Fig. 4(b), Fig. 5(b) and Fig. 6(b), it can be seen that in the three experiments, the solution obtained by the BSO-LNS algorithm is the smallest and requires only a small number of iterations to converge to the optimal solution, and the variance of the results of the multiple experiments is 0 or very close to 0. This indicates that the performance and robustness of the

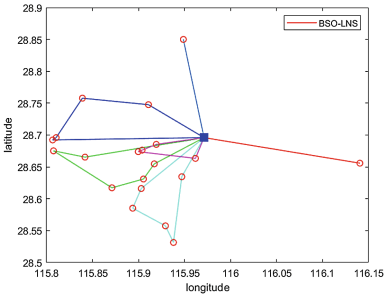


(a) BSO-LNS algorithm optimal solution

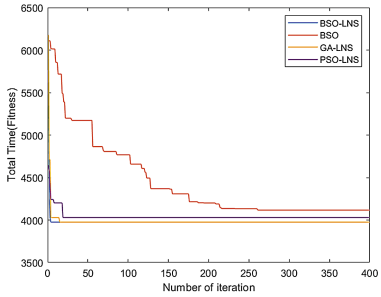


(b) Performance comparison

Fig. 5. Experiment 2.



(a) BSO-LNS algorithm optimal solution



(b) Performance comparison

Fig. 6. Experiment 3.

BSO-LNS algorithm are very good, and compared with GA and PSO, the BSO is more suitable to be mixed with LNS algorithm to solve VRPTW.

Many experiments have found that the total delivery time of Experiment 2 is higher than that of Experiment 1, which shows that the road conditions under Experiment 2 conditions are more congested than Experiment 1, indicating that road conditions have an impact on urban distribution efficiency. The optimal solution of Experiment 3 is very different from Experiment 2, and combined with Fig. 5(a) and Fig. 6(a), BSO-LNS only needs to arrange 4 vehicles in Experiment 2, while the same distribution data needs to arrange 6 vehicles in Experiment 3, which shows that the traditional calculation of straight-line distance without considering road conditions does not have great reference significance for distribution in real scenarios.

From Table 6, it can be found that the solution obtained by the original BSO algorithm in Experiment 1 will have the situation that the actual vehicle load exceeds the rated vehicle load, such as route $0 \rightarrow 16 \rightarrow 10 \rightarrow 1 \rightarrow 8 \rightarrow 0$ is a violation of the loading capacity constraint (the maximum vehicle load is 7

Table 6. Random result of experiment 1.of 10 runs.

Alg	Distribution route	Loading(t)	Loading rate (%)	Value
BSO-LNS	0-12-19-6-9-15-7-3-0	6.311108	90.15869	Time of use: 461 min
	0-2-11-1-10-8-0	6.903115	98.61593	
	0-16-18-5-0	5.94997	84.99957	
	0-20-14-17-0	6.938785	99.1255	
	0-4-13-0	6.48	92.57143	
GA-LNS	0-14-3-7-6-18-5-0	6.623931	94.62759	Time of use: 464 min
	0-16-17-0	5.58634	79.80486	
	0-2-11-10-1-8-0	6.903115	98.61593	
	0-20-12-19-4-0	6.49778	92.82543	
	0-9-15-13-0	6.971812	99.59731	
BSO	0-20-14-17-0	6.938785	99.1255	Time of use: 484 min
	0-3-9-15-7-11-2-0	6.094927	87.07039	
	0-19-6-18-5-0	5.52997	78.99957	
	0-16-10-1-8-0	7.049296	100.7042	
	0-12-4-13-0	6.97	99.57143	
PSO-LNS	0-11-10-1-2-0	6.349599	90.70855714	Time of use: 470 min
	0-12-19-6-9-15-7-3-0	6.311108	90.15868571	
	0-20-14-17-0	6.938785	99.1255	
	0-8-16-18-5-0	6.503486	92.90694286	
	0-4-13-0	6.48	92.57142857	

tons), which is undesirable in real life. This can be avoided by mixing LNS in the BSO algorithm, which makes the global search and local search capability of the hybrid algorithm further enhanced.

5 Conclusion

In this paper, based on the real time road conditions, we abandon the traditional VRPTW, which is often calculated by the straight line distance between two points and assuming a constant speed, and use a web crawler to automatically obtain the actual passage time of vehicles between two points from AMap. The efficient hybrid BSO-LNS algorithm is used for the solution and the results are compared with PSO-LNS, GA-LNS and the original BSO algorithm. The obtained results are better than the other three algorithms and are more robust. The experimental results also show that the impact on realistic urban distribution is significant in the case of considering road conditions and not considering road conditions. In future work, we will consider acquiring more real-time road condition data and taking travel time dynamics into account with time series prediction.

Acknowledgment. The work described in this paper was supported by Natural Science Foundation of Guangdong Province (Grant No. 2020A1515010749), Key Research Foundation of Higher Education of Guangdong Provincial Education Bureau (Grant No. 2019KZDXM030), University Innovation Team Project of Guangdong Province (Grant No. 2021WCXTD002).

References

1. Solomon, M.M., Desrosiers, J.: Survey papertime window constrained routing and scheduling problems. Transp. Sci. **22**(1), 1–13 (1988)
2. Jabali, O., Leus, R., Van Woensel, T., De Kok, T.: Self-imposed time windows in vehicle routing problems. Or Spectrum **37**(2), 331–352 (2015)
3. Berov, T.D.: A vehicle routing planning system for goods distribution in urban areas using google maps and genetic algorithm. Int. J. Traffic Transp. Eng. **6**(2), 159–167 (2016)
4. Liu, J., Hu, X., Chen, J., Chen, X., Wen, X.: Research on urban distribution optimization under point-based billing on simulated annealing with variable neighborhood. J. Uncertain Syst. **15**(01), 2250005 (2022)
5. Wang, J., Pu, K., Shen, Z.: Urban distribution vehicle routing optimization and empirical analysis under the influence of carbon trading policy. In: 2018 3rd International Conference on Politics, Economics and Law (ICPEL 2018), pp. 411–416. Atlantis Press (2018)
6. Zheng, W., Wang, Z., Sun, L.: Collaborative vehicle routing problem in the urban ring logistics network under the Covid-19 epidemic. Math. Probl. Eng. **2021** (2021)
7. Leng, K., Li, S.: Distribution path optimization for intelligent logistics vehicles of urban rail transportation using VRP optimization model. IEEE Trans. Intell. Transp. Syst. **23**(2), 1661–1669 (2021)
8. Taniguchi, E., Shimamoto, H.: Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times. Transp. Res. Part C Emerg. Technol. **12**(3–4), 235–250 (2004)
9. Kritzinger, S., Doerner, K.F., Hartl, R.F., Kiechle, G., Stadler, H., Manohar, S.S.: Using traffic information for time-dependent vehicle routing. Procedia-Soc. Behav. Sci. **39**, 217–229 (2012)
10. Falek, A.M., Gallais, A., Pelsser, C., Julien, S., Theoleyre, F.: To re-route, or not to re-route: impact of real-time re-routing in urban road networks. J. Intell. Transp. Syst. **26**(2), 198–212 (2022)
11. Yu, G., Yang, Y.: Dynamic routing with real-time traffic information. Oper. Res. **19**(4), 1033–1058 (2019)
12. Liu, C., Kou, G., Zhou, X., Peng, Y., Sheng, H., Alsaadi, F.E.: Time-dependent vehicle routing problem with time windows of city logistics with a congestion avoidance approach. Knowl.-Based Syst. **188**, 104813 (2020)
13. Shaw, P.: A new local search algorithm providing high quality solutions to vehicle routing problems, p. 46. APES Group, Department of Computer Science, University of Strathclyde, Glasgow, Scotland, UK (1997)
14. Shi, Y.: Brain storm optimization algorithm. In: Tan, Y., Shi, Y., Chai, Y., Wang, G. (eds.) ICSI 2011. LNCS, vol. 6728, pp. 303–309. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21515-5_36
15. Ke, L.: A brain storm optimization approach for the cumulative capacitated vehicle routing problem. Memetic Comput. **10**(4), 411–421 (2018)

16. Song, M.X., Li, J.Q., Han, Y.Y., Zheng, Z.X.: Solving the vehicle routing problem with time window by using an improved brain strom optimization. In: 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 1306–1313. IEEE (2019)
17. Wu, L., He, Z., Chen, Y., Wu, D., Cui, J.: Brainstorming-based ant colony optimization for vehicle routing with soft time windows. IEEE Access **7**, 19643–19652 (2019)
18. Shen, Y., Liu, M., Yang, J., Shi, Y., Middendorf, M.: A hybrid swarm intelligence algorithm for vehicle routing problem with time windows. IEEE Access **8**, 93882–93893 (2020)
19. Liang, X., Yang, J., Xiang, Z., Chen, Y.: Two-stage brain storm optimization-simulated annealing algorithm for constrained vehicle routing problem. In: 2021 3rd International Academic Exchange Conference on Science and Technology Innovation (IAECST), pp. 1357–1362. IEEE (2021)
20. Liu, M., Shen, Y., Zhao, Q., Shi, Y.: A hybrid BSO-ACS algorithm for vehicle routing problem with time windows on road networks. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2020)