Int j simul model 18 (2019) 4, 708-719
ISSN 1726-4529 Original scientific paper

SPLIT-DELIVERY VEHICLE ROUTING PROBLEMS BASED ON A MULTI-RESTART IMPROVED SWEEP APPROACH

Min, J. N.[*]; Jin, C.[*,**,#] & Lu, L. J.[*,***]
[*] School of Economics and Management, Taihu University of Wuxi, Wuxi 214064, China
[**] School of Management, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
[***] School of Management, Nanjing University, Nanjing 210093, China
E-Mail: mjn3862@126.com, jcm3988@126.com, dg1702501@smail.nju.edu.cn
([#] Corresponding author)

Abstract
The vehicle routing problem (VRP) with split delivery is an important branch of the classic VRP. The objective is to reduce the transportation distance and number of vehicles used. This study proposes a two-stage algorithm based on an improved sweep heuristic approach to solve this problem. We cluster customer points into a minimum number of groups through a multi-restart-iteration sweeping algorithm (MRISA). The load demands and split point in each group are fine-adjusted using the load rate and threshold coefficient. To reduce the travel distance in each group, an optimal route is produced by a tabu search algorithm (TSA). The computational simulation results demonstrate that the proposed method in this study is feasible and efficient, and the near-optimal performance of the proposed method is achieved regarding the transportation distance and computation time to the instances with a scattered distribution geographical characteristic; however, the strategy of "the maximum-minimum distance clustering first, and MRISA + TSA executing later" is much more useful for instances with a clustered-distribution geographical characteristic.
(Received, processed and accepted by the Chinese Representative Office.)

Key Words: Fine-Tuning, Multi-Restart Improved Sweep Algorithm, Tabu Search, VRP

1. INTRODUCTION

In 1989, Dror and Trudeau [1] first presented the split-delivery vehicle routing problem (SDVRP), in which the delivery of a quantity to a point can be split into several routes and fulfilled by multiple vehicles or many times of one vehicle [1, 2]. Thus, the VRP can be optimized in terms of number of vehicles used and transportation distance. Therefore, the SDVRP has rapidly become an important research area of the classic VRP [3].

Heuristics, meta-heuristics, multi-stage hybrid heuristics, and exact solution methods have been proposed for the SDVRP [3]. Dror and Trudeau [1, 2] investigated the local-search algorithm for the SDVRP by assuming that each point demand was equal to or less than Q ($d_i \leq Q$). The algorithm comprises two phases. The first phase is k-split, where a customer point with its demand is split into several routes in which the remaining capacity is sufficient for this split customer. The second phase is route creation, where a split point is removed from all routes where it stays and a new route comprising only this split point is generated. The experimental results demonstrate that the major savings are caused by the high delivery quantities.

Archetti et al. [4] studied the first tabu search algorithm, called SPLITABU, considering the following three phases. First, an initial feasible solution is constructed, where a travelling salesman problem (TSP) on all customers was generated and this large tour was cut into pieces to satisfy the vehicle capacity limitation. Then, in the tabu search algorithm, two procedures of ordering routes and selection of the best neighbourhood were performed to obtain the best routes observed to date. At last, GENIUS was adopted to reduce the length of each route to further improve the solution. Numerical experiments proved that this algorithm is much more effective than Dror and Trudeau’s algorithm [1, 2], although it is obviously much slower. Jin et al. [5] proposed a two-stage algorithm with valid inequalities. In the first stage, they divided the customer points into groups and solved the TSP for each group to determine the minimum travel distance. In the second stage, the sum of the minimum travel distances over all groups was used to update the objective function, and the procedure was iterated. The algorithm can then solve instances with up to 22 vertices, but with considerable computational effort. Chen et al. [6] discussed the first hybrid heuristic algorithm, where the initial solution was obtained using the Clarke-Wright saving algorithm for the VRP. Then, an endpoint mixed integer program (EMIP) was applied to this initial solution to optimally reallocate the endpoints of each route. Using a variable length record-to-record travel algorithm, the solution obtained from EMIP was improved. Based on the same set of instances, even when the CPU time tends to be considerably high for large instances, the computational results illustrated that this approach outperformed TSA in [4].

Aleman et al. [7] proposed a two-stage approach. First, an initial solution was generated using a constructive algorithm, where a variable neighbourhood descent program was first used to improve the initial solution; then, numerical tests were conducted on the instances in [4-6, 8]. Aleman and Hill [9] developed a vocabulary construction method for tabu search. In addition to the search, the set of solutions was constantly changing; here, a good solution was added to the collection when a bad solution was removed. The approach clearly improved the results in [7]. Gulczynski et al. [10] considered split deliveries only if a minimum fraction of customer demand was serviced by a vehicle, and developed an EMIP with an enhanced record-to-record travel algorithm to solve this problem. Archetti et al. [11] conducted a branch-price-and-cut algorithm based on the strategy of dividing the problem into sub-problems, i.e., routes. A column corresponded to a route with delivery quantities. The set of columns was applied to search for an optimal solution for SDVRP. The experimental results illustrated that the algorithm could obtain better solutions for most of the tested instances. Based on this algorithm, a commodity-constrained SDVRP, comprising 40 customers, each of which had three commodities, was solved in [12].

Liu et al. [13] constructed a *k*-means clustering algorithm and designed two types of solutions: (1) grouping first and routing later and (2) routing first and grouping later. Their experimental results illustrated that the former performed better. Wilck and Cavalier [14] studied a two-stage clustering first and routing later heuristic algorithm for solving the SDVRP. There were three operations in the clustering. First, the outermost point from the depot was selected, and then an initial path from the depot to this point was formed. Then, the average remaining capacity was rounded up to the nearest integer. Finally, the nearest neighbour was gradually included in the constructed initial path until the sum of point demands in the path was equal to the vehicle capacity. The clustering ended when no point was free outside the abovementioned paths. The calculations demonstrated that it performed better based on the same dataset regarding the travelled distance and calculated speed. Wang et al. [15] formulated a bee colony optimization model for the SDVRP depending on the reaction threshold and stimulatory value. The experimental results indicated the feasibility of the algorithm. Wen [16] presented a three-stage local-search algorithm. First, GENIUS was adopted to solve the TSP path for the whole customer domain; all points in the path were partitioned into groups based on the vehicle capacity. Then, for each point, two operations, i.e., deletion from its current path and reinsertion into an optimal position using a point-inserting greedy algorithm based on the split demand strategy, were iterated until no more improvement occurred. Finally, through the disturbance, the two operations of deletion and reinsertion were repeated until the optimal solution was reached. The experimental results showed that the algorithm could obtain better results.

Xiang and Pan [17] investigated a two-stage approach with the strategy of "routing after clustering." The "nearest" tactics were applied in the clustering stage. The ant colony optimization algorithm was adopted to schedule the routes. Shi and Zhang [18] proposed a model for the SDVRP with stochastic customers. Here, modified split insertion operators and a large adaptive neighbour search heuristic were considered. Luo et al. [19] discussed a branch-price-cut approach for the SDVRP with time windows and linear weight-related cost. Tang et al. [20] developed a three-stage simulation approach, based on the non- deterministic polynomial model to minimize the total travel distance of cars, to find key factors affecting the location of electric vehicle charging stations.

Ozbaygin et al. [21] studied an exact flow-based formulation using vehicle indexes. By aggregating the decision variables over all vehicles, the size of the vehicle-indexed formulation was reduced via a relaxation procedure. The optimal solutions could be obtained either by locally extending the formulation using the vehicle-indexed variables or by node splitting. Shi et al. [22] considered a local search-based particle swarm approach. They designed coding and decoding methods as well as best-position vectors to optimize the SDVRP solutions. Balamurugan et al. [23] studied a two-stage inventory routing problem to minimize the total amount of carbon dioxide emission by reducing the total distance travelled by all the vehicles. The evolution was based on the induction of centralized collection and distribution strategy in inventory management and the adoption of an artificial immune algorithm.

Note that the heuristic methods employed in the aforementioned studies are generally tedious and most of the exact algorithms developed can solve the SDVRP only at a small scale. To solve these problems based on the largest possible vehicle capacity fine-tuned by an appropriate load rate (LR) and the threshold coefficient (TC), this paper proposes a multi-restart improved sweep algorithm (MRISA) to partition all customer points into groups. We adopted a TSA to optimize the route in each group. In this study, the computing simulations we conducted on the benchmark datasets demonstrate that the proposed method (MRISA + TSA) is feasible and efficient, and can obtain optimal approximate solutions in a few seconds to the instances with a scattered-distribution geographical characteristic. Moreover, in this study, a three-stage approach of "maximum-minimum distance (Max-Min dis)" + MRISA + TSA can provide optimal approximate solutions in a few seconds to the instances with a clustered-distribution geographical characteristic.

The rest of this paper is organized as follows. In Section 2, the SDVRP is described. In Section 3, the proposed two-stage method is introduced. In Section 4, the simulation results are presented and discussed. Finally, in Section 5, the conclusions are presented.

2. PROBLEM MODELLING

In the SDVRP, a vehicle departs from the depot, serves the customers with delivery demands, and finally returns to the depot. Any number of visits is allowed when vehicles serve customers; each customer can be visited by one or more different vehicles multiple times. The aim is to obtain a set of vehicle paths that reduces the total travelling distance and a number of vehicles used to the lowest possible number under the vehicle-capacity limitation.

In the considered SDVRP, assume that there are n customers and m vehicles to be used. The SDVRP is an un-digraph $G = (V, E)$, where $V|V \ni (0, 1, 2, \ldots, n)$ denotes the vertex set and E denotes the edge set. A customer point is indexed by $i|\, i = (1, 2, \ldots, n)$ and $i = 0$ indicates the depot. A vehicle is expressed as $v \mid v = (1, 2, \ldots, m)$. Note that vehicles in a fleet are of the same model. The maximum load/capacity of a vehicle is denoted by Q. A vehicle departs from the depot, delivers the customers' demands along one route, and then returns to the depot without any load. The delivery demand of customer i is denoted by d_i, while the delivery amount between customers *i* and *j* is denoted by d_{ij}. The distance between customers i and j is denoted by c_{ij} ($c_{ii} = 0, c_{ij} = c_{ji}$, non-negative, and satisfies triangular inequality). The minimum number of vehicles to be utilized is given by $\lceil \sum_{i=1}^{n} d_i/Q \rceil$ [2], where $\lceil x \rceil$ denotes the smallest integer not less than x. A customer's requirements should be fully satisfied. The objective function is applied to schedule appropriate routes such that the total travel distance of vehicles is the shortest.

The following are the decision variables:

$$x_{ij}^{v} = \begin{cases} 1, & \text{vehicle } v \text{ goes from } i \text{ to } j; \\ 0, & \text{others;} \end{cases} \qquad i, j = (0, 1, 2, ..., n);\ v = (1, 2, ..., m)$$

$$y_{iv} = \begin{cases} y_{iv}, & \text{the delivery quantity of } i \text{ by vehicle } v; \\ 0, & \text{others;} \end{cases} \qquad i, j = (0, 1, 2, ..., n);\ v = (1, 2, ..., m)$$

The objective function is formulated as follows:

$$min \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{v=1}^{m} c_{ij}\, x_{ij}$$

Subject to the following constraints:

$$\sum_{i=0}^{n} \sum_{v=1}^{m} x_{ij}^{v} \geq 1, j = 0, 1, \dots, n;\ v = 1, 2, \dots, m \tag{1}$$

$$\sum_{i=0}^{n} x_{ip}^{v} - \sum_{j=0}^{n} x_{pj}^{v} = 0, p = 0, 1, \dots, n; v = 1, 2, \dots, m \tag{2}$$

$$\sum_{i\in S} \sum_{j\in S} x_{ij}^{v} \leq |S| - 1, v = 1, 2, \dots, m; SV - \{0\} \tag{3}$$

$$y_{iv} \leq d_i \sum_{j=0}^{n} x_{ij}^{v}, i = 1, 2, \dots, n; v = 1, 2, \dots, m \tag{4}$$

$$\sum_{v=1}^{m} y_{iv} = d_i, i = 1, 2, \dots, n \tag{5}$$

$$\sum_{i=1}^{n} y_{iv} \leq Q, v = 1, 2, \dots, m \tag{6}$$

Eq. (1) indicates that each point must be served at least once. Eq. (2) illustrates the flow-conservation restraint. Eq. (3) gives the sub-route elimination restraint. Eq. (4) demonstrates that vehicle v serves point i only if point i is visited by vehicle v. Eq. (5) guarantees that all customer requirements are satisfied. Eq. (6) ensures that the load of a vehicle does not go beyond its largest capacity.

3. A TWO-STAGE APPROACH

The proposed method in this study is a two-stage approach based on the strategy of "routing after grouping." The first stage involves using an MRISA to divide the customers into clusters and determining the split points in each partition. The second stage involves route optimization: a TSA is applied to determine the optimal route for each cluster. The proposed algorithm, MRISA + TSA, is detailed in the following subsections.

3.1 Pre-process

The deliveries of each point, $d_i > Q$, should be considered at the beginning. The weight, Q, can be directly carried by a vehicle, and the remaining points, $d_i = (d|i - Q)$, are analysed using the following procedure.

3.2 MRISA

In the 1970s, Gillett and Miller [24] proposed the sweep algorithm that essentially partitions the nearest customers into a group. The classic sweep algorithm is modified to satisfy the requirements of the SDVRP. $\lceil \sum_{i=1}^{n} d_i/Q \rceil$ is utilized to determine the number of groups, and multi-restart iteration is harnessed to determine the last and split points for each group. Both LR and TC are applied to fine-tune the group partitions. The procedure of MRISA is detailed as follows.

Step 1: Transform the coordinate systems.

a) Create a polar coordinate system: Define the depot as the original point of this polar coordinate system.
b) Set angle 0: Create a line between the original point and one of the customer's points, and set it to be angle 0.
c) Transform: Convert all points to this system.
d) Sort: Sort all angles in an ascending order.

Step 2: Partition the customer domain

Step 2.1: Create the variables *InitialValue* and *StoragePool*, and employ the former and the latter to store the initial and the best value, respectively.

Step 2.2: First loop

a) Set start point: Set the point of angle 0 as the first starting point.
b) Sweep: Sweep the points one-by-one into this group in a clockwise (or anticlockwise) manner until the cumulative value (*CV*) of $\sum_{i=1}^{n} d_i$ exceeds Q.
c) Create a group: Split this last point lp into lp_1 and lp_2; lp_1 makes the load of the current group, d_{lp1}, to be equal to Q. This group ends with lp_1.
d) Create the next group: Start the next group from point lp_2 (with demand d_{lp2}).
e) End the sweeping: Repeat Step 2.2 (b-d) until the last point is swept.
f) Compute the travelling distance: Calculate the total transportation distance based on the spatial distribution of the points in each cluster.
g) Store the travelling distance: Place the total transportation distance into *InitialValue* and *StoragePool*.

Step 2.3: Restart iterations

Step 2.3.1: Sweep clockwise

a) Iterate: Sequentially set each point as the starting point in a clockwise manner.
b) Continue: Execute Step 2.2 (b-f).
c) Optimize: Compare the current total travel distance with the value in *StoragePool*. If the value in *StoragePool* is higher, it is replaced with the current one.
d) Repeat: Execute Step 2.3.1 in an ascending order until the last point is reached.

Step 2.3.2: Sweep anticlockwise

a) Iterate: In an anticlockwise manner, sequentially set each point as the starting point.
b) Continue: Execute Step 2.3.1 (b and c).

c) Repeat: Execute Step 2.3.2 in a descending order until the last point is reached.

Step 2.3.3: Apply *LR* and *TC*

a) Fine-tuning using *LR*: *LR* changes Q to $LR \cdot Q$ to balance each group load.

b) Fine-tuning using *TC*: Refine Step 2.2 (b) into multiple control branches with *TC* (e.g., set $TC = 2$ or $TC = 4$), which is used to further fine-adjust the partition. The control branches are described as follows.

- If $CV < LR \cdot Q$ cumulate the next i value.
- If $CV = LR \cdot Q$ end the current group and start the next group.
- If $CV > LR \cdot Q$ and $CV < Q$, cumulate the next i value if $(d_i - (CV - LR \cdot Q)) \leq TC \cdot (1 - LR) \cdot Q$; split the last point, end the current group, and start the next group if $(d_i - (CV - LR \cdot Q)) > TC \cdot (1 - LR) \cdot Q$.
- If $CV < Q$, end the current group and start the next group.
- If $CV > Q$ split the last point, end the current group, and start the next group.

Step 3: Output the results:

a) Output the total travel distance in *InitialValue* and *StoragePool*.

b) Output points, split points and the corresponding split values in each cluster.

3.3 Route optimization

As mentioned in Sections 3.1 and 3.2, the problem domain is divided into several smaller clusters (one route in a cluster), and a TSA is used to optimize the route in each cluster.

In 1986, Glover proposed a TSA, which is a meta-heuristic. It is a dynamic neighbourhood search algorithm whose procedure is as follows.

Step 1: Initialization-set the crucial variables:

a) Set *tabuLength* <– the length of tabu.

b) Set *maxIter* <– the maximum number of iterations.

c) Set *Tabu*[*tabuLength*] <– a tabu list with the *tabuLength*.

d) Set *currentbestQueue*[*customerNum*] <– generate the initial route randomly.

Step 2: Calculate the current objective function

a) Set *bestValue* <– the value of the objective function of the *currentbestQueue* [*customerNum*] (be able to briefly as *currentbestQueue* [*cst-Num*]).

Step 3: Conditional judgment

a) If the number of iteration $t = MaxIter$, stop the program and export the optimal results.

b) Otherwise, repeat constantly and implement the following steps.

Step 4: Generate the neighbourhoods

a) Set *neighbourRR.listNeighbour*[*rr*][*customerNum*] <– generate *rr* neighbours of the *currentbestQueue*[*cst-Num*] (e.g., 2-opts).

b) Set *neighbourRR.obDist*[*rr*] <– calculate each neighbour's objective function in *neighbourRR.listNeighbour*[*rr*][*customerNum*].

c) Sort *neighbourRR*[*rr*][*customerNum*] in a non-descending order based on *neighbourRR.obDist*[*rr*].

Step 5: Decide

a) If *neighbourRR.obDist*[0] < *bestValue,* set:

- *bestValue* <– *neighbourRR.obDist*[0].
- *currentbestValue* <– *neighbourRR.obDist*[0].
- *bestQueue*[*customerNum*] <– *neighbourRR.listNeighbour*[0][*customerNum*].
- *currentbestQueue*[*cst-Num*]<– *neighbourRR.listNeighbour*[0][*cst-Num*].
- Go to Step 7.

b) Otherwise, execute Step 6.

Step 6: Analyse *neighbourRR*[*rr*], set:

```
        a) currentbestValue <– neighbourRR.obDist[t]
                                   // the best value of neighbourRR.obDist [rr].
        b) currentbestQueue[cst-Num]<– neighbourRR.listNeighbour[t][ cst-Num]
                                   //the corresponding object of the neighbourRR.obDist[t].
Step 7: Iterate:
        a) Set the number of iterations + 1.
Step 8: Go to Step 3.
```

4. SIMULATION AND RESULTS ANALYSIS

In this study, computational simulations were conducted to verify the feasibility and effectiveness of the proposed method. Two benchmark datasets from the capacitated VRP (CVRPLIB) in the VRP web were adopted. Case 1 is from the Christofides and Eilon benchmark dataset, while Case 2 is from the Christofides, Mingozzi, and Toth benchmark dataset. The experiments were conducted in *C* using a 64-bit computer with an Intel (R) Core processor 2.50 GHz and 8 GB memory.

4.1 Case 1

There are 11 instances in Case 1. The computational simulation results for the instances are presented in Table I. The instance name "E-n21-k4-22500-6000" indicates "Eil dataset, 21 points, 4 routes, 22500 customer demands, and 6000 vehicle capacity". For each instance, two operations are executed: clockwise and anticlockwise (briefly as anti-). In each operation, the results for two control branches are recorded for each instance: the *LR* control, *TC* = 2, and *TC* = 4 control. The *LR* control indicates that only *LR* is used for fine-tuning in MRISA; *TC* = 2 or *TC* = 4 control indicates that both *LR* and *TC* = 2 or *TC* = 4 control is used for fine-tuning. In each control branch column, the three sub-columns indicate *LR*, the initial value generated in the 0^{th} loop, and the best value of MRISA.

As shown in Table I, for each instance (Int.), we have the following: (1) The travelling distances (*m*) vary during the operation (Opt.) in either anticlockwise or clockwise. (2) The best value (Best) is always lesser than the corresponding initial value (Init.). (3) The last best value can appear in either anticlockwise or clockwise operation shown in green colour. These results demonstrate that MRISA is necessary, feasible, and effective. (4) The *RDP*s of instances are all less than 8 %, and 7 out of 11 are less than 5 %, where:

$$RDP = ((Dis_{MRISA+TSA} - Dis_{reference\ data}) / Dis_{reference\ data}) \cdot 100\,\%.$$

We observed that instances 6-9 in Table I have the same point positions and demands, but with different vehicle capacities and number of routes. The number of routes (*Rt.*) and the distance (*Dis.*) in each instance increases as the vehicle capacity *Q* decreases. Moreover, the range of the ratio of each point demand to *Q* (Ratio) increases as the vehicle capacity *Q* decreases, as shown in the column of "split" in the upper part (in instances 1-4) of Table II.

To further simulate the influence of the ratio of demand to *Q* on the number of routes and travel distance, we further reduce *Q*. Therefore, the ratio of demand to *Q* is further increased, and the number of routes and distance are further increased sharply, as shown in instances 5-6 in the lower part of Table II. Thus, when the positions and demands are the same and the vehicle capacity keeps decreasing, the travel distance and number of vehicles to be used increases.

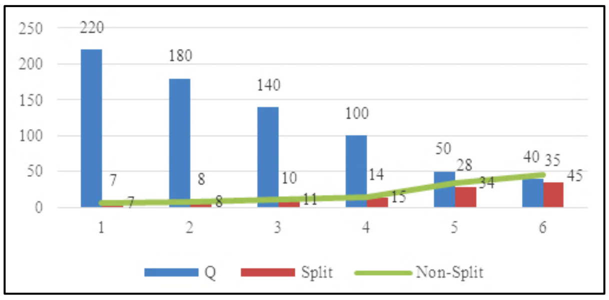
When a non-split operation is adopted, the number of routes increases, the distance increases, and *LR* decreases. The changes are shown in Fig. 1 a-d. These results demonstrate that the splitting delivery is necessary and is efficient in decreasing the travel distance and number of vehicles used.

Table I: Simulation results of Case 1.

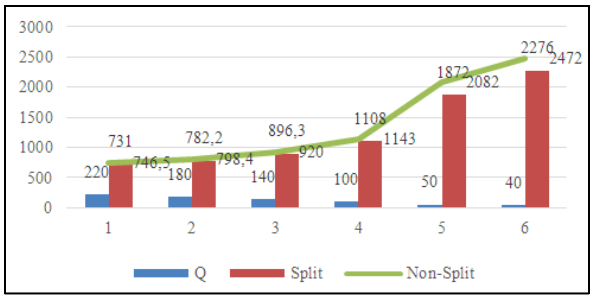
	Int.	**Opt.**	***LR* control**			***TC* = 2 control**			***TC* = 4 control**			***RDP***
			LR	**Init.**	**Best**	***LR***	**Init.**	**Best**	***LR***	**Init.**	**Best**	(%)
1	E-n21-k4-22500-6000	Anti-	0.95	433.2	408.7	0.88	410.5	398.6	0.90	413.2	381.14	1.64
		Clockwise	0.98	431.5	409.4	0.90	431.5	413.2	0.90	431.5	413.2	
2	E-n22-k3-101890-4500	Anti-	0.70	766.2	605.9	0.70	766.2	587.2	0.70	738.73	570.3	0.23
		Clockwise	0.70	768.2	570.3	0.65	763.9	587.2	0.65	763.9	587.2	
3	E-n29-k3-12750-4500	Anti-	0.95	605.6	564.7	0.90	605.6	565.3	0.92	573.8	563.7	
		Clockwise	0.95	705.2	559.8	0.95	589.8	559.8	0.94	566.8	517.6	-3.07
4	E-n32-k4-29370-8000	Anti-	0.95	881.4	881.4	0.90	881.4	881.4	0.90	881.4	881.4	
		Clockwise	0.95	910.7	881.4	0.82	875.6	875.6	0.82	875.6	868.2	3.98
5	E-n50-k5-777-160	Anti-	0.95	604.6	569.1	0.95	609.6	569.1	0.95	609.6	553.0	6.14
		Clockwise	0.95	610.3	569.1	0.96	610.3	569.9	0.96	610.3	569.9	
6	E-n75-k7-1364-220	Anti-	0.90	801.5	774.7	0.86	789.5	731.7	0.86	789.5	716.46	4.90
		Clockwise	0.90	825.8	774.1	0.88	780.1	750.5	0.88	780.1	750.5	
7	E-n75-k8-1364-180	Ant-	1.0	863.4	810.7	0.92	809.6	782.2	0.92	809.6	764.39	4.0
		Clockwise	0.95	827.4	799.5	0.92	817.6	798.4	0.92	817.6	798.4	
8	E-n75-k10-1364-140	Anti-	0.95	941.3	895.4	0.93	941.3	896.3	0.93	941.3	896.3	
		Clockwise	0.95	904.6	887.0	0.93	904.6	893.2	0.93	896.3	875.39	5.22
9	E-n75-k14-1364-100	Anti-	1.0	1172	1109	0.94	1178	1114	0.93	1169	1097	
		Clockwise	1.0	1174	1108	0.94	1097	1103	0.93	1173	1095	6.11
10	E-n100-k8-1458-200	Anti-	0.92	920.7	900.8	0.90	899.3	877.4	0.90	899.3	877.4	
		Clockwise	0.92	933.9	891.3	0.90	891.2	875.5	0.90	891.2	857.27	4.93
11	E-n100-k14-1458-112	Anti-	0.88	1226	1166	0.88	1244	1177	0.88	1217	1162	7.89
		Clockwise	0.88	1209	1180	0.88	1209	1191	0.88	1209	1180	

Table II: Changes in other items with corresponding changes in *Q*.

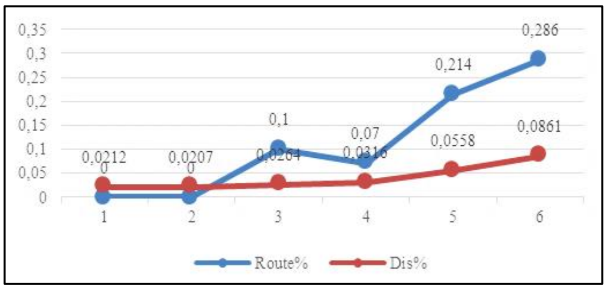
	Instance	***Q***	**Ratio**	**Split**			**Non-Split**				
				Rt.	***Dis.***	***LR***	***Rt.***	**Δ*Rt.***	***Dis.***	**Δ*Dis.***	***LR***
1	E-n75-k7-1364-220	220	0.00455–0.168	7	731.0	0.89	7	0	746.5	0.0212	0.89
2	E-n75-k8-1364-180	180	0.00556–0.206	8	782.2	0.95	8	0	798.4	0.0207	0.95
3	E-n75-k10-1364-140	140	0.00714–0.264	10	896.3	0.97	11	0.1	920	0.0264	0.89
4	E-n75-k14-1364-100	100	0.01–0.37	14	1108	0.97	15	0.07	1143	0.0316	0.91
5	E-n75-k28-1364-50	50	0.02–0.74	28	1872	0.97	34	0.214	2082	0.0558	0.80
6	E-n75-k35-1364-40	40	0.025–0.925	35	2276	0.97	45	0.286	2472	0.0861	0.76



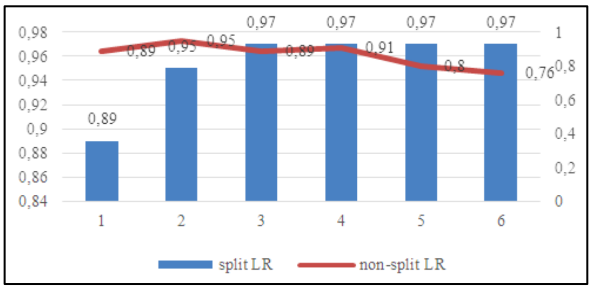
a) Changes in routes with *Q*



b) Changes in distances with *Q*



c) Changes in routes and distances



d) Changes in load rates

Figure 1: Various changes in routes, distances and load rates.

4.2 Case 2

There are six instances in Case 2. The computational results for the instances of Case 2 are presented in Table III. The instance name "v1-50-5" indicates "vrpnc1 dataset, 50 points, and 5 routes". Based on Case 2, Table III presents the simulation results of SPLITABU, NSA + TSA, and MRISA + TSA. In each method, the total travel distance (*Dis*.) and computing time (*T*) are recorded, and *RDP* (1) of the travel distance of each method against MRISA + TSA is calculated.

$$RDP\ (1) = ((Dis_{MRISA+TSA} - Dis_{selected\ algorithm}) / Dis_{selected\ algorithm}) \cdot 100\,\%.$$

Table III shows the following: (1) *RDP* (1) of travel distances between SPLITABU and MRISA + TSA except for instances 6-7 are all less than 6 %, and MRISA + TSA can obtain near-optimal solutions to instances with a scattered-distribution geographical characteristic of instances 1-5. (2) *RDP* (1) of travel distances between NSA + TSA and MRISA + TSA are all less than 0. The results indicate that the travel distances of MRISA + TSA are less than that of NSA + TSA, and MRISA is necessary, feasible, and effective. (3) The computation time of MRISA + TSA is much lower than that of SPLITABU. The largest amount of computation time is equal to 15 s.

Table III: Simulation results of Case 2.

	Int.	SPLITABU				NSA+TSA				MRISA+TSA			MaxMin	
		Dis.	*T*	*RDP* (1) %	*RDP* (2) %	*Dis.*	*T*	*RDP* (1) %	*RDP* (2) %	*Dis.*	*T*	*RDP* (2) %	*Dis.*	*T*
1	v1-50-5	528	17	4.86		578	0.6	-4.3		553	1.4			
2	v2-75-10	854	64	4.63		922	2.0	-3.2		893	2.9			
3	v3-100-8	840	60	5.49		908	2.9	-2.4		886	3.5			
4	v4-150-12	1055	440	4.17		1128	7.0	-2.5		1099	7.3			
5	v5-199-16	1338	1900	4.25		1444	12.6	-3.4		1395	15			
6	v11-120-7	1057	39	20.9	6.9	1372	3.9	-6.9	-17.6	1278	6.6	-11.6	1130	7.8

Taking instances 1 as examples, the route graph is shown in Fig. 2 where the red lines indicate the routes shared by two adjacent routes, the red points express the split points, while the red square represents the depot.

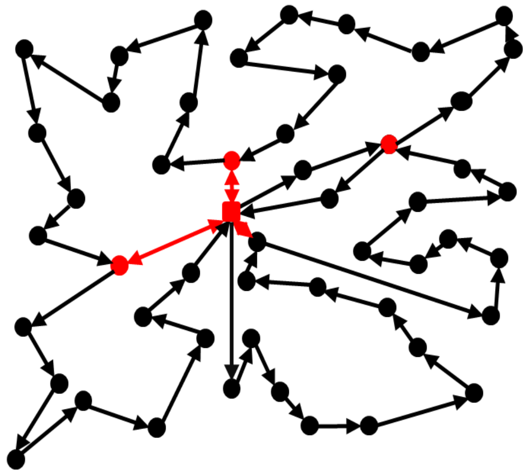


Figure 2: Routes in instance 1 with scattered-distribution points.

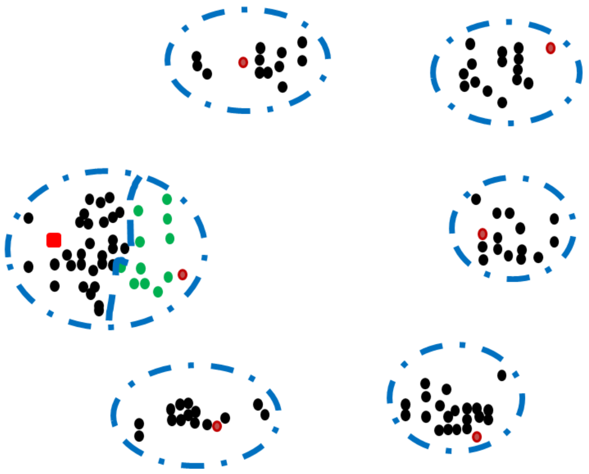


Figure 3: Instance 6 with clustered-distribution points.

Furthermore, Table III demonstrates that *RDP* (1) of SPLITABU of instance 6 "v11-120-7" is much larger. This indicates that MRISA is not efficient in this instance. The characteristic of the geographical distribution of instance "v11-120-7" is that all points are distributed as clusters around the depot shown in Fig. 3 (the red square indicates the depot and the dark red points indicate the clustering centres). To fit this feature, the strategy of "clustering first, and MRISA + TSA later" is adopted. First, the Max-Min dis clustering method [25, 26] is adopted to cluster the customer's domain. The clusters are shown as dotted ellipses in Fig. 3 (the most left dotted ellipse comprises two clusters separated by a dotted curve line and points in different colours). Then, MRISA + TSA is used in each cluster. It is a three-stage approach of "Max-Min dis" + MRISA + TSA.

The computational result is shown in the column "Max-Min dis" + MRISA + TSA (briefly as MaxMin). The result demonstrates that the travelling distance of MaxMin, which is painted yellow, is reduced. *RDP* (2) of travel distances with SPLITABU, NSA + TSA against MRISA + TSA are shown in column *RDP* (2) of the corresponding algorithms. The *RDPs* of SPLITABU and NSA + TSA decreased by about 14 % (from 20.9 to 6.9) and 10.8 % (from -6.9 to -17.6), respectively.

The simulation results indicate that the three-stage approach of "Max-Min dis" + MRISA + TSA is much more useful than MRISA + TSA for instances with such a clustered-distribution geographical characteristic.

5. CONCLUSION

This study proposed a two-stage method based on an MRISA constructive heuristic for the SDVRP. In the first stage, a multi-restart iteration strategy was employed to sweep the customer domain in both anticlockwise and clockwise directions and to start the operations at each point. The customer domain was divided into sub-domains based on the vehicle capacity. *LR* and *TC* were employed to fine-adjust the partitions, split points, and split loads. In the second stage, a TSA was applied to optimize a route in each sub-domain for achieving the minimum total transportation distance.

The computing simulation utilizing the benchmark datasets were conducted. The computational results demonstrated that the multi-restart iteration strategy was necessary, and the method of MRISA + TSA provided feasible and effective solutions in most instances of the test datasets. The approximate optimal solutions were obtained as *RDP* < 5 % in 7 out of 11 instances in dataset 1 and *RDP* < 6 % in 4 out of 5 instances in dataset 2. Moreover, the largest amount of computation time was less than 3 s in dataset 1 and 15 s in dataset 2, both of which were significantly shorter than those in the other algorithms evaluated.

Additional simulations demonstrated that the MRISA + TSA method is efficient for the instances with a scattered-distribution geographical characteristic, while the three-stage approach of "Max-Min dis" + MRISA + TSA is much more useful than MRISA + TSA for instances with a clustered-distribution geographical characteristic.

ACKNOWLEDGEMENT

This work is sponsored by National Natural Science Foundation of China (Grant No. 61872077), the Natural Science Fund of Jiangsu Province Education Commission (Grant No. 17KJB520040), the Humanities and Social Sciences Research Base Fund of Jiangsu Province Education Commission (Grant No. 2017ZSJD020), and the Jiangsu Key Construction Laboratory of IoT Application Technology, Taihu University of Wuxi.

REFERENCES

[1] Dror, M.; Trudeau, P. (1989). Savings by split delivery routing, *Transportation Science*, Vol. 23, No. 2, 141-145, doi:10.1287/trsc.23.2.141

[2] Dror, M.; Trudeau, P. (1990). Split delivery routing, *Naval Research Logistics*, Vol. 37, No. 3, 383-402, doi:10.1002/nav.3800370304

[3] Archettic, C.; Speranza, M. G. (2013). Vehicle routing problems with split deliveries, *International Transactions in Operational Research*, Vol. 19, Vol. 1-2, 3-22, doi:10.1111/j.1475-3995.2011.00811.x

[4] Archettic, C.; Speranza, M. G.; Hertz, A. (2006). A tabu search algorithm for the split delivery vehicle routing problem, *Transportation Science*, Vol. 40, No. 1, 64-73, doi:10.1287/trsc.1040.0103

[5] Jin, M.; Liu, K.; Bowden, R. O. (2007). A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem, *International Journal of Production Economics*, Vol. 105, No. 1, 228-242, doi:10.1016/j.ijpe.2006.04.014

[6] Chen, S.; Golden, B.; Wasil, E. (2007). The split delivery vehicle routing problem: applications, algorithms, test problems, and computational results, *Networks*, Vol. 49, No. 4, 318-329, doi:10.1002/net.20181

[7] Aleman, R. E.; Zhang, X.; Hill, R. R. (2010). An adaptive memory algorithm for the split delivery vehicle routing problem, *Journal of Heuristics*, Vol. 16, No. 3, 441-473, doi:10.1007/s10732-008-9101-3

[8] Belenguer, J. M.; Martinez, M. C.; Mota, E. (2000). A lower bound for the split delivery vehicle routing problem, *Operations Research*, Vol. 48, No. 5, 801-810, doi:10.1287/opre.48.5.801.12407

[9] Aleman, R. E.; Hill, R. R. (2010). A tabu search with vocabulary building approach for the vehicle routing problem with split demands, *International Journal of Metaheuristics*, Vol. 1, No. 1, 55-80, doi:10.1504/IJMHEUR.2010.033123

[10] Gulczynski, D.; Golden, B.; Wasil, E. (2010). The split delivery vehicle routing problem with minimum delivery amounts, *Transportation Research Part E: Logistics and Transportation Review*, Vol. 46, No. 5, 612-626, doi:10.1016/j.tre.2009.12.007

[11] Archetti, C.; Bianchessi, N.; Speranza, M. G. (2011). A column generation approach for the split delivery vehicle routing problem, *Networks*, Vol. 58, No. 4, 241-254, doi:10.1002/net.20467

[12] Archetti, C.; Bianchessi, N.; Speranza, M. G. (2015). A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem, *Computers & Operations Research*, Vol. 64, 1-10, doi:10.1016/j.cor.2015.04.023

[13] Liu, W.-S.; Yang, F.; Li, M.-Q.; Chen, P.-Z. (2012). Clustering algorithm for split delivery vehicle routing problem, *Control and Decision*, Vol. 27, No. 4, 535-541, doi:10.13195/j.cd.2012.04.57.liuwsh.017

[14] Wilck IV, J. H.; Cavalier, T. M. (2012). A construction heuristic for the split delivery vehicle routing problem, *American Journal of Operations Research*, Vol. 2, No. 2, 153-162, doi:10.4236/ajor.2012.22018

[15] Wang, T.-T.; Ni, Y.-D.; He, W.-L. (2014). Bee colony optimization algorithm for split delivery vehicle routing problem, *Journal of Hefei University of Technology*, Vol. 2014, No. 8, 1015-1018, doi:10.3969/j.issn.1003-5060.2014.08.024

[16] Wen, Z. Z. (2015). *Researches on iterated local search for the split delivery vehicle routing problem*, Beijing Transportation University, Beijing

[17] Xiang, T.; Pan, D. (2016). Clustering algorithm for split delivery vehicle routing problem, *Journal of Computer Applications*, Vol. 36, No. 11, 3141-3145, doi:10.11772/j.issn.1001-9081.2016.11.3141

[18] Shi, J.-L.; Zhang, J. (2017). Model and algorithm for split delivery vehicle routing problem with stochastic customers, *Control and Decision*, Vol. 32, No. 2, 213-222, doi:10.13195/j.kzyjc.2016.0065

[19] Luo, Z.; Qin, H.; Zhu, W.; Lim, A. (2016). Branch and price and cut for the split-delivery vehicle routing problem with time windows and linear weight-related cost, *Transportation Science*, Vol. 51, No. 2, 668-687, doi:10.1287/trsc.2015.0666

[20] Tang, M.; Gong, D.; Liu, S.; Lu, X. (2017). Finding key factors affecting the locations of electric vehicle charging stations: a simulation and ANOVA approach, *International Journal of Simulation Modelling*, Vol. 16, No. 3, 541-554, doi:10.2507/IJSIMM16(3)CO15

[21] Ozbaygin, G.; Karasan, O.; Yaman, H. (2018). New exact solution approaches for the split delivery vehicle routing problem, *EURO Journal on Computational Optimization*, Vol. 6, No. 1, 85-115, doi:10.1007/s13675-017-0089-z

[22] Shi, J.; Zhang, J.; Wang, K.; Fang, X. (2018). Particle swarm optimization for split delivery vehicle routing problem, *Asia-Pacific Journal of Operational Research*, Vol. 35, No. 2, paper 1840006, doi:10.1142/S0217595918400067

[23] Balamurugan, T.; Karunamoorthy, L.; Arunkumar, N.; Santhosh, D. (2018). Optimization of inventory routing problem to minimize carbon dioxide emission, *International Journal of Simulation Modelling*, Vol. 17, No. 1, 42-54, doi:10.2507/IJSIMM17(1)410

[24] Gillet, B. E.; Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem, *Operations Research*, Vol. 22, No. 2, 340-349, doi:10.1287/opre.22.2.340

[25] Min, J.; Jin, C.; Lu, L. (2018). A three-stage approach for split delivery vehicle routing problem solving, *Proceedings of the 8th International Conference on Logistics, Informatics and Service Sciences*, 1-6, doi:10.1109/LISS.2018.8593226

[26] Min, J. N.; Jin, C.; Lu, L. J. (2019). Maximum-minimum distance clustering method for split delivery vehicle routing problem: case studies and performance comparisons, *Advances in Production Engineering & Management*, Vol. 14, No. 1, 125-135, doi:10.14743/apem2019.1.316