APEM journal
Advances in Production Engineering & Management
Volume 14 | Number 1 | March 2019 | pp 125–135
https://doi.org/10.14743/apem2019.1.316
ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

Maximum-minimum distance clustering method for split-delivery vehicle-routing problem: Case studies and performance comparisons

Min, J.N.[a], **Jin, C.**[a,*], **Lu, L.J.**[a,b]

[a]Taihu University of Wuxi, School of Economics and Management, Jiangsu, P.R. China
[b]Nanjing University, School of Management, Nanjing, Jiangsu, P.R. China

ABSTRACT

The split-delivery vehicle-routing problem in which delivery to a demand point can be served by any number of vehicles is an important branch of classic VRP. Objective function is used to minimise travel distance while using the lowest number of vehicles. According to the maximum-minimum distance clustering method, a three-stage algorithm is proposed. First, the maximum-minimum distance method is employed to cluster customer points into the lowest number of groups. Second, according to the maximum vehicle capacity, the load demand in each group is adjusted to create suitable customer points in each clustering group by adopting 'push-out' and 'pull-in' operations. Third, a tabu search is used and an optimised route for each group is generated to minimise the total travel distance. Numerical experiments, some on the benchmark data set, are presented to verify the feasibility and effectiveness of the proposed algorithm. The computational results show that the performance of the proposed algorithm is better in terms of both optimised travel distance and less computation time when the problem size is less than 75. The results also show that when the customer points are in a cluster distribution around the depot, the algorithm achieves better performance.

ARTICLE INFO

Keywords:
Split-delivery vehicle-routing problem;
Maximum-minimum distance method;
Load-demand adjustment;
Route optimisation;
Tabu search;
Clustering first and routing later

**Corresponding author:*
mjn3862@126.com
(Jin, C.)

Article history:
Received 5 December 2018
Revised 12 February 2019
Accepted 25 February 2019

1. Introduction

The split-delivery vehicle-routing problem (SDVRP) was formally introduced by Dror and Trudeau [1] in 1989. In SDVRP, the constraint in which each customer is visited only once, which is imposed in the classic VRP, is relaxed [2]. In other words, in SDVRP, delivery to a demand point can be split among any number of vehicles [1-3]. This relaxation can further optimise the VRP in terms of the number of vehicles used, travel distance, and carbon emissions, which have recently become a significant environmental protection problem. The fractional capacity of a vehicle can be better utilised. When the remaining capacity of the vehicle is insufficient to provide complete service to a customer, that vehicle can still provide service to the customer that is equal to the residual capacity, and the remaining customer demand can be served by other vehicles. The transportation cost can be reduced because of the following reasons. (1) The optimal solution of the number of vehicles used in SDVRP might be lower than that used in the VRP. (2) Largest savings are obtained when the average customer demand is only more than half of the vehicle capacity, and the variance in the customer demands is low [4, 5]. Therefore, the SDVRP has quickly

become an important branch of VRP since it has been introduced and has received increasing attention over time [2].

Studies have mainly focused on attempting to solve the problem in designing heuristic and meta-heuristic solution approaches [2]. Dror and Trudeau [1] proposed a first heuristic algorithm for the SDVRP. Two types of methods were introduced: the first method is the *k*-split interchange, i.e. a customer demand is split into different routes in which the remaining capacity is sufficient for the split customer. The second method is route addition, i.e. a split customer is removed from all the routes where the customer lies and a new route is created, which is composed of that customer only. The computational results showed that main savings are achieved for large values of customer demands. Archetti *et al.* [4] proposed the first tabu search algorithm (TSA) called SPLITABU. At each iteration, a neighbour solution is obtained by removing a customer from a set of routes in which the customer is currently visited and inserting the customer into either a new route or an existing route with sufficient residual capacity. A numerical experiment proved that this is much more effective than the algorithm of Dror and Trudeau [1] even though it is obviously much slower. Chen *et al.* [5] proposed the first hybrid heuristic algorithm. In this algorithm, the initial solution is obtained using the Clarke-Wright saving algorithm for the VRP. Then, an endpoint mixed-integer program (EMIP) is applied to this initial solution to optimally reallocate the endpoints of each route. The solution obtained from EMIP is then improved using a variable-length record-to-record travel algorithm. The computational results showed that the performance of this approach is better than that of the TSA in [4] under the same set of instances even if the CPU time tends to be quite high in large instances. Archetti *et al.* [6] proposed an improved three-stage algorithm based on the TSA proposed in [4]. First, the SDVRP is solved by applying the TSA in [4]. Then, the solutions are analysed to ascertain particular information, namely, (1) if an edge is often traversed in these solutions, then this edge is included in high-quality solutions with a high probability, and (2) if a customer is never or rarely split in the TSA solution, then that customer is probably not split in high-quality solutions. Once the TS ends, the information from (1) is used to reduce the original graph by discarding those edges that are never or rarely traversed by the TSA. All possible routes are constructed based on this reduced graph. The information from (2) is used to construct a route-based formulation to identify the best routes among the set of generated routes. Finally, the mixed-integer linear programming (MILP) model is repeatedly applied on small subsets of routes in the ascending order, which is sorted according to the 'desirable' parameter. The computational results of this algorithm improved those produced under the same set of instances presented by the TSA in [4] in many cases. Gulczynski *et al.* [7] allowed split deliveries only if a minimum fraction of a customer demand was serviced by a vehicle and developed an EMIP with an enhanced record-to-record travel algorithm to solve this problem. Wilck IV and Cavalier [8] developed a construction heuristic algorithm to solve the SDVRP, and their computational results showed that its performance is better than the two-phase method under the same data set in terms of travel distance and computation speed. Liu *et al.* [9] proposed the *k*-means clustering algorithm and designed two different algorithmic approaches: one was the grouping first and routing later; the other was the routing first and grouping later. A comparison of these two approaches indicated that the approach of grouping first and routing later exhibited better performance. Lu *et al*. [10] proposed a routing optimisation algorithm for different electric-vehicle-movement situations. A multi-agent simulation model was run using city real data. Wang *et al.* [11] proposed a bee-colony optimisation model for the SDVRP based on the reaction threshold and stimulatory value. The experimental results indicated the feasibility of the algorithm. Zhu *et al.* [12] addressed a multi-depot capacitated VRP where the client demand is composed of two-dimensional weighted items. A quantum-behaved particle swarm optimisation and an exploration heuristic local search algorithm were proposed, and computational experiments on the benchmark instances were effectively carried out. Wen [13] proposed a multi-re-start iteration local search (MRSILS) algorithm. First, a large travelling salesman problem (TSP), which includes all customer points, is solved by adopting GENIUS. The solution is partitioned into groups according to the vehicle capacity in order for each group to meet the load-demand limitations. Then, for each point, a greedy point re-insertion algorithm is used to delete a point from its current group and

re-insert it into an optimal position in the current solution by considering the delivery-split tactics. This re-insertion is iterated until no further improvement occurs. Finally, the 'perturbation' and solution pool are adopted to repeat the re-insertion in order to obtain the optimisation result. The experimental results on the benchmark data set showed that the MRSILS is competitive. Wu *et al.* [14] introduced a multi-objective algorithm to solve VRPs using time windows. The algorithm combines a discrete particle swarm optimisation based on a set-decoding scheme and variable neighbourhood searches to find the Pareto optimal routing solutions. Tang *et al.* [15] developed a model and the corresponding multi-phase particle swarm optimisation algorithm for bulk-cargo port scheduling. Xiang *et al.* [16] proposed a clustering algorithm of 'routing after grouping'. The grouping is based on the 'nearest' principle. A split threshold is set to limit the vehicle load to within a certain range, and the ant-colony optimisation algorithm is used to arrange the routes. Cao *et al.* [17] proposed an improved wolf-pack algorithm to solve the VRP using multiple fuzzy-time windows. Johanyák *et al.* [18] proposed a modified particle swarm optimisation algorithm, which was combined with a local-search technique, to solve large-scale nonlinear optimisation problems. Wang *et al.* [19] adopted a hybrid fruit-fly optimisation algorithm integrated with three local search methods (two-opt, swap, and insert) to solve the multi-compartment VRP.

Research in the SDVRP solution also extends to exact algorithms [2]. Belenguer *et al.* [20] proposed a cutting-plane approach wherein a polyhedron model was built. Some facet-inducing and other valid inequalities were embedded in the cutting-plane algorithm. They solved a relatively smaller problem and achieved satisfactory results. Lee *et al.* [21] proposed the shortest-path approach in which the SDVRP is formulated as a dynamic programming model, where the routes are sequentially constructed using a labelling algorithm. The solution space and corresponding states were reduced according to the *k*-split-cycle property. They tested their approach on instances with up to seven customers. Jin *et al.* [22] proposed a two-stage algorithm with valid inequalities (TSVI). The first stage divided the customers into clusters and established a lower bound. The objective function minimised the clustering cost in which the cost of each cluster was initially set to zero. The second stage calculated the minimum travel distance in each cluster by solving the corresponding TSP. The sum of the minimum distance travelled over all clusters yielded an upper bound. The minimum distance of each TSP was used to update the objective function in the first stage, and the procedure was iterated. Valid inequalities were developed and added to strengthen the MILP model. The approach was able to solve instances with up to 22 vertices but with a large computational effort. Archetti *et al.* [23] implemented a branch-price-cut algorithm based on the principle of problem decomposition. Each column generated by the sub-problem represented a route with delivery quantities. The generated columns were used to find an optimal heuristic solution to the problem. Both cases where the fleet of vehicles was unlimited and limited to the minimum possible number of vehicles were considered. The computational results showed that the algorithm reduced the optimality gap in most of the benchmark instances. Archetti *et al.* [24] proposed a branch-price-cut algorithm for a commodity-constrained SDVRP. They solved up to 40 customers with three commodities per customer. Luo *et al.* [25] proposed branch, price, and cut for the SDVRP with time windows and linear weight-related cost. Ozbaygin *et al.* [26] proposed an exact flow-based formulation using vehicle indexes. The size of this vehicle-indexed formulation was reduced via a relaxation procedure by aggregating the decision variables over all vehicles. The optimal solutions could be obtained either by locally extending the formulation using the vehicle-indexed variables or by node splitting.

Most techniques used in this research field are heuristic and meta-heuristic methods. These are time-consuming methods because multiple iterations or comparisons of many results are necessary to find the optimal solution. Meanwhile, most of the exact algorithms provided in the literature can only solve small-size SDVRPs. To solve these time-consuming and large-scale problems, we propose an algorithm of 'clustering first and routing later' [27]. It consists of firstly clustering the domain of the customer points according to the 'nearest' principle, adjusting the load demands for each sub-domain according to the maximum vehicle-load capacity, and finally routing each sub-domain to minimise the travel distance. In this manner, a near-optimal solution can be obtained in less time. To obtain more optimised results, the modified version of the first

two stages are provided, and TS is used in the third stage in this study. More case studies on the benchmark data set are used, and the computational results are compared to verify the feasibility and effectiveness of the algorithm proposed in this study. The proposed algorithm can be of practical value for both reducing the time consumption and shortening the travel distance.

The remainder of this paper is organised as follows. In Section 2, the SDVRP is described. In Section 3, a three-stage algorithm based on the 'clustering first and routing later' strategy plus TS (CRTS) for the SDVRP is presented. In Section 4, the computational results are shown and discussed. Finally, the conclusion is presented in Section 5.

2. Problem description

SDVRP is undigraph $G = (V, E)$, where V is the vertex set, i.e. $V = \{0,1, \dots, m\}$. 0 stands for the depot, and the other vertices stand for the customer points. E is the edge set. c_{ij} is the length of edge (i, j) $(c_{ij} \in E)$; it is non-negative and satisfies the triangle inequality. d_i stands for the demand of customer point i, where $i \in V - \{0\}$. The vehicles in the fleet are homogenous, and the maximum load capacity of a vehicle is Q. The lowest vehicle number is $\lceil \sum_{i=1}^{n} d_i / Q \rceil$ [2]. Each vehicle starts from and ends at the depot. The customer demands should be completely satisfied. The carrying weight of a vehicle in each route cannot be larger than Q. The following notations are used.

x_{ij}^v is equal to one when vehicle v directly moves from i to j; otherwise, $x_{ij}^v = 0$. y_{iv} is the quantity of the demand of i delivered by vehicle v.

The objective function is to minimise the total travel distances of the vehicles and is formulated as follows:

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

s. t.

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

$$\sum_{i=0}^{n} x_{ip}^v - \sum_{j=0}^{n} x_{pj}^v = 0, p = 0,1,2, \dots, n; v = 1,2, \dots, m \tag{3}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij}^v \leq |S| - 1, v = 1,2, \dots, m; S \subseteq V\text{-}\{0\} \tag{4}$$

$$y_{iv} \leq d_i \sum_{j=0}^{n} x_{ij}^v, i = 1,2, \dots, n; v = 1,2, \dots, m \tag{5}$$

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

$$\sum_{i=1}^{n} y_{iv} \leq Q, v = 1,2, \dots, m \tag{7}$$

$$x_{ij}^v \in \{0,1\}, i, j = 0,1, \dots, n; v = 1,2, \dots, m \tag{8}$$

$$y_{iv} \geq 0, i = 1,2, \dots, n; v = 1,2, \dots m. \tag{9}$$

Eq. 2 states that each customer point can be visited at least once. Eq. 3 is the flow-conservation constraint. Eq. 4 is the sub-route elimination constraint. Eq. 5 states that customer point i is served by vehicle v only if vehicle v visits customer point i. Eq. 6 ensures that all customer demands are met. Eq. 7 ensures that each vehicle does not exceed its maximum load capacity.

3. Used methods: Proposed algorithm

The proposed algorithm is a three-stage algorithm based on the CRTS. The first stage clusters the customers. The maximum-minimum distance method (Max-Min dis) is adopted to cluster the customer points according to their geographic locations. The second stage adjusts the load weight for each cluster. The 'push out' and 'pull in' operations are used to adjust the load weight to collect all possible points in each cluster according to the load demand. The 'push out' operation directs the extra weight away from a cluster, and the 'pull in' operation directs some demands from the nearest customer points of neighbour clusters into clusters whose weights are less than Q. The third stage optimises the route, which uses the TSA. The details of the proposed algorithm are described in the next sections.

3.1 Pre-processing

Load demand of each customer point $d_i > Q$ should be handled before the CRTS. Weight Q can be transported by one vehicle, and remaining weight $d_i = d_i - Q$ at this point is handled using the following procedures.

3.2 Clustering the customers

Basic idea of the maximum-minimum distance

Max-Min dis is a type of pattern recognition method, which can improve the efficiency of dividing the initial data set [28].

Procedure of the Max-Min dis

Step 1: Take x_1 (normally x_1 is the depot) as first clustering centre z_1.
Step 2: Calculate distances D_{i1} at each point i to z_1, $i = 1,2, \dots, n$.
Step 3: Take x_k as second clustering centre z_2 if $D_{k1} = max\{D_{i1}\}$.
Step 4: Calculate distances D_{i1} and D_{i2} at each point i to z_1 and z_2.
Step 5: Take x_l as third clustering centre z_3; if $D_l = \max\{\min(D_{i1}, D_{i2})\}$ and $D_l > \theta \cdot D_{12}$, D_{12} is the distance of z_1 and z_2, where θ is in [0,1], a selectable parameter to meet the requirement of $[\sum_{i=1}^{n} d_i/Q]$.
Step 6: Calculate $D_j = \max\{\min(D_{i1}, D_{i2}, D_{i3})\}$ and $D_j > \theta \cdot D_{12}$; if z_3 exists, x_j is taken as fourth clustering centre z_4 and so on until $D_j \leq \theta \cdot D_{12}$; finally, end the procedure.
Step 7: Classify all points into the nearest clusters according to the principle of minimum distance.

3.3 Adjusting the load weight for each cluster

After clustering the customer points, the calculation and adjustment of the load weight for each cluster is performed as follows.

'Push out' procedure

Step 1: If load weight w_g of a cluster is in $[\alpha \cdot Q, Q]$ and the number of points in this cluster is two, then these two points form a route. If μ_1 clusters similar to this exist, then μ_1 routes are created.
Step 2: If load weight w_g of a cluster is in $[Q, 2 \cdot Q]$ and the number of points in this cluster is two, then these two points form an inner route, and load $w_g - Q$ is pushed out. If μ_2 clusters similar to this exist, then μ_2 routes are created.
Step 3: If load weight w_g of a cluster is larger than $\eta \cdot Q (\eta > 2)$, then $(\eta - 1)$ inner routes are formed in this cluster, and the remaining demand $w_g - (\eta - 1) \cdot Q$ is pushed out. If μ_3 clusters similar to this exist, then $\sum_{i=1}^{\mu_3}(\eta_i - 1)$ routes are created. After the abovementioned three steps are completed, clustering of the remaining customer points in these clusters proceeds to Step 4.
Step 4: The remaining customer points are re-clustered to form $[\sum_{i=1}^{n} d_i/Q] - \mu$, $\mu = \mu_1 + \mu_2 + \mu_3 * (\eta - 1)$ clustering groups, where α is in $[\sum_{i=1}^{n} d_i/([\sum_{i=1}^{n} d_i/Q] * Q)$, 1].

'Pull in' procedure

Step 1: The clusters with less than Q are visited according to the cluster number sequence, e.g. the first one is group $A(w_{gA} < Q)$.
Step 2: Among the neighbours, nearest point t to centre i of cluster A is searched, e.g. cluster B.
Step 3: If point t has load weight $d_t > (Q - w_{gA})$ and the load weight of cluster B w_{gB} is larger than Q, then point t is split into t and t', $d_{t'} = Q - w_{gA}$ is moved to cluster A, and $d_t = d_t - (Q - w_{gA})$ is maintained.
Step 4: If point t has demand $d_t = (Q - w_{gA})$ and the load weight of cluster B w_{gB} is larger than Q, then point t is moved (merged) into cluster A.
Step 5: If point t does not have sufficient load demand $d_t < (Q - w_{gA})$, then point t is first moved (merged) into cluster A. Second, point t', which is nearest to t in cluster B, is searched if w_{gB} is sufficient; otherwise, point t' is searched in cluster C of the next nearest neighbour from cluster A. Third, Step 3 is performed.
Step 6: If the load weight of group A is in $[\alpha \cdot Q, Q]$, then handling for cluster A is terminated. If the load weight of group A $w_{gA} < \alpha \cdot Q$, then Step 2 is performed.
Step 7: If all groups are visited, the procedure is terminated; otherwise, Step 1 is performed.

3.4 Optimising the routes

The problem domain has been divided into several smaller-sized clusters after the abovementioned three operations. Thus, many algorithms can be used to optimise the solution. We adopt the TSA to optimise the route in each cluster. The detailed procedure for the TS is described as follows.

Step 1: Initialising the variables
Set: *tL* (tabu length) = T, *MIs* (terminal condition maximum iterations) = NG,
cN (customerNum) = N
Create *tabu* [*tL*];
Generate randomly *serialNum* [*cN*] (initial route solution)
Step 2: Calculate the objective function value *serialNum* [*cN*] and deposit it into the variable *bestValue*
Step 3: If iteration = NG, terminate the program and output the optimal results; otherwise, continuously iterate and execute the following steps in each loop.
Step 4: Generate *rr* neighbourhood of the current solution by adopting suitable selection functions (e.g. two-opts).
Step 5: Sort it according to a non-descending order and store it into variable *tempDist* [*rr*].
Step 6: If *tempDist* [0] < *bestValue*, let (assumed at the m^{th} iteration):
bestValue = *tempDist* [0]; *currentBestValue* = *tempDist* [0]
bestQueue = the corresponding objects of *tempDist*[0]
currentBestQueue of NG_m = the corresponding objects of *tempDist*[0]
tabu = corresponding objects of *tempDist*[0].
Go to Step3
Otherwise, execute the following steps:
Step 7: Analyse the tabu attributes of the corresponding objects of *tempDist* [*rr*]
currentBestValue = the best value of *tempDist* [*i*] (assume the i^{th} one)
currentBestQueue of NG_m = the corresponding objects of *tempDist*[*i*]
Step 8: Go to Step3.

4. Case studies

To verify the feasibility and effectiveness of the proposed algorithm, three case studies were adopted. Case study 1 was from [22] and was compared with its TSVI algorithm. Case study 2 was from [16] and was compared with its clustering algorithm with a splitting threshold and the *k*-means clustering algorithm in [9], which are all based on the 'routing after grouping' strate-

gy. Case study 3 involved the benchmark data set from the Capacitated Vehicle Routing Problem library (CVRPLIB) and was compared with the SPLITABU algorithm in [4] and MRSILS in [13]. The numerical experiments were implemented on *C* in a Windows 7 64-bit machine with an Intel (R) Core processor and 8 GB of memory.

4.1 Case study 1

Case study 1 contains three instances (N7L1-N7L3). The number of clients in each instance is $N = 7$, the maximum load of the vehicle is one, and the coordinates of the depot are (0, 0). In Table 1, the columns indicate the total travel distance (*Dis.*), consumed computational (CPU) time (*T*) of the TSVI (the data come from [22]) and CRTS, and the relative deviation percentage (*RDP*) of the distance between the CRTS and TSVI. *RDP* (*Dis.*) is equal to $((Distance_{CRTS} - Distance_{algorithm})/Distance_{algorithm})\cdot 100\,\%$, and *RDP* (*T*) is equal to $((Time_{CRTS} - Time_{algorithm})/Time_{algorithm})\cdot 100\,\%$.

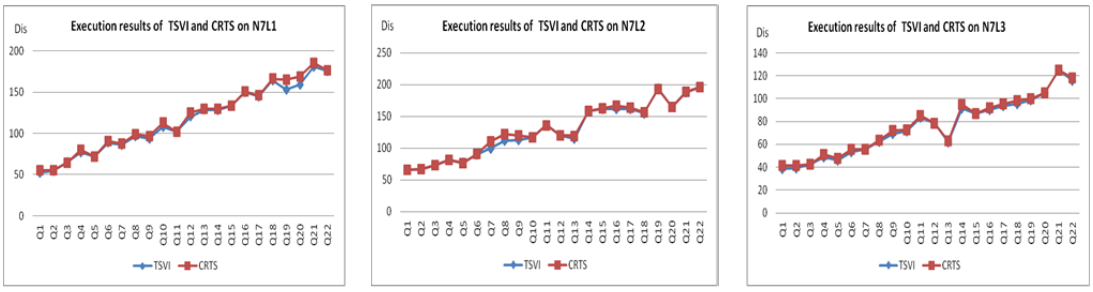
The executions in which *RDP* was equal to or less than 1.0 % comprised 48.49 % of the total 66 executions. The executions in which *RDP* was larger than 5.0 % were 12.12 % of the total 66 executions.

The CPU time (*T*) of CRTS listed in Table 1 is equal to 100 times the actual value (to save display space), which indicates that the consumed CPU time of the CRTS is much lower than that of the TVSI.

The results of the total travel distance obtained by the CRTS and TSVI in the three instances in case study 1 are shown in Fig. 1. The good consistency of the curves of the two different algorithms indicates that the CRTS algorithm is feasible and effective.

Table 1 Comparisons of the results between the CRTS and TSVI

No.	N7L1 execution results					N7L2 execution results					N7L3 execution results				
	TSVI		CRTS		RDP	TSVI		CRTS		RDP	TSVI		CRTS		RDP
	Dis.	T	Dis.	T	%	Dis.	T	Dis.	T	%	Dis.	T	Dis.	T	%
Q1	52.33	<1	55.34	0.2	5.75	65.48	<1	66.04	0.2	0.86	38.35	<1	41.22	0.2	7.48
Q2	54.47	<1	55.34	0.2	1.6	66.70	<1	66.70	1.2	0	39.21	<1	41.22	0.2	5.13
Q3	64.67	<1	64.16	1.2	-0.79	73.02	<1	73.02	1.2	0	42.60	<1	42.6	1.4	0
Q4	77.27	2	79.75	1.1	3.21	81.14	<1	81.70	1.2	0.69	48.89	<1	51.18	1.1	4.68
Q5	71.86	<1	71.87	1.3	0.01	77.34	<1	76.44	1.2	-1.16	45.95	<1	47.66	1.1	3.72
Q6	88.67	1	90.34	0.2	1.88	90.11	<1	91.01	0.2	1.0	53.14	<1	55.62	1.1	4.67
Q7	85.80	<1	87.33	1.1	1.78	99.76	1	110.61	1.1	10.88	55.62	<1	55.62	1.1	0
Q8	96.76	1	98.79	1.4	2.1	111.74	1	122.37	1.4	9.51	62.45	<1	63.31	1.7	1.38
Q9	93.46	<1	96.35	1.1	3.09	112.52	<1	120.34	1.3	6.95	69.24	<1	71.97	1.2	3.94
Q10	107.60	2	112.76	1.3	4.8	116.85	<1	116.91	1.6	0.05	71.39	2	72.63	1.4	1.74
Q11	101.79	<1	101.79	0.2	0	136.10	<1	136.10	0.2	0	83.52	<1	85.18	1.2	1.99
Q12	120.26	2	124.86	1.2	3.83	120.04	<1	120.04	1.1	0	78.59	<1	78.59	1.2	0
Q13	128.50	23	129.54	1.3	0.81	114.39	1	119.56	1.3	4.52	61.92	<1	63.14	1.3	1.97
Q14	128.15	<1	129.38	1.4	0.96	158.24	<1	158.55	1.4	0.2	91.37	<1	94.78	1.2	3.73
Q15	133.13	2	133.18	1.3	0.04	161.42	1	162.69	1.2	0.79	86.84	1	86.84	1.2	0
Q16	149.70	3	150.53	0.2	0.55	161.46	1	166.79	1.1	3.3	90.37	3	91.82	1.2	1.6
Q17	144.97	<1	145.80	0.2	0.57	161.91	1	163.85	1.2	1.2	93.89	1	95.24	1.0	1.44
Q18	164.07	2	166.12	1.4	1.25	154.89	1	156.32	1.2	0.92	95.13	<1	97.97	13	2.99
Q19	153.07	1	165.20	1.2	7.92	193.60	3	192.71	1.5	-0.46	99.02	5	99.90	1.2	0.89
Q20	159.19	7	169.07	1.2	6.21	164.49	10	164.81	1.1	0.19	105.11	5	105.11	1.2	0
Q21	180.87	1	185.27	1.2	2.43	188.13	8	189.16	1.8	0.55	125.13	1	125.12	1.1	0
Q22	175.68	8	176.48	1.3	0.46	196.13	2	195.83	1.1	-0.15	116.02	2	118.20	1.1	1.88



N7L1 N7L2 N7L3

Fig. 1 Computational results of the three instances in case study 1

4.2 Case study 2

Case study 2 contains three instances. The basic information of the customer number, total customer demand, capacity of vehicles, and vehicle number required in each instance are listed in Table 2.

In Table 3, the columns indicate the total travel distance (*Dis.*), consumed CPU time (*T*) of the *k*-means clustering algorithm, the clustering algorithm with splitting threshold and the CRTS (the former two data come respectively from [9] and [16]), and *RDP* of the distance and time between the CRTS and each of the above-described algorithms. A positive *RDP* value indicates that the CRTS value is large, whereas a negative *RDP* value indicates that the CRTS value is small.

Table 3 lists the following. (1) From the perspective of the total travel distance, the clustering algorithm with a split threshold is the lowest in the two instances of $N = 15$ and $N = 20$, and the *RDP* (*Dis.*) values are 0.815 % and 4.067 % lower than those of the CRTS, respectively. Meanwhile, the CRTS at $N = 36$ is the lowest, and the *RDP* (*Dis.*) values are 0.805 % and 1.25 % lower than the resolutions of the other two algorithms, respectively. (2) From the perspective of the consumed CPU time, the CRTS consumes the lowest CPU time.

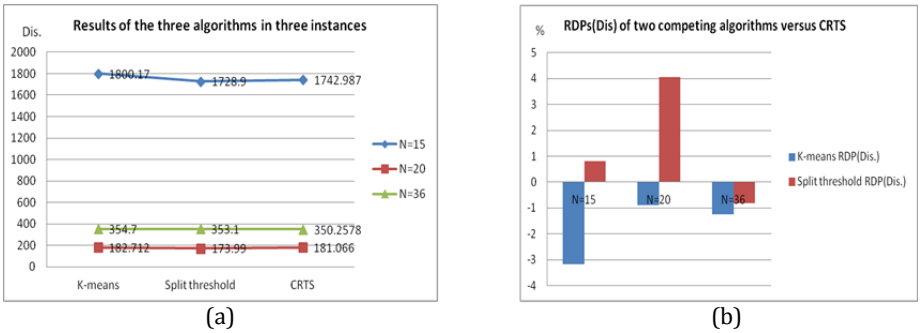
Fig. 2 clearly shows that the CRTS is feasible and effective. The curves in Fig. 2(a) show that the gap in the total travel distances of the three algorithms in each of the three instances, namely, $N = 15$, $N = 20$, and $N = 36$, is relatively small. The CRTS obtains better results. Fig. 2(b) shows that the CRTS performance is between that of the *k*-means and split-threshold approaches in terms of the total travel distance.

Table 2 Basic information of each instance in case study 2

Instance	Customer number	Total demand	Capacity of vehicles	Vehicle number
2-1	15	4,881	500	10
2-2	20	40	5	8
2-3	36	15.29	1	16

Table 3 Computational results of the three algorithms

N	*k*-means				Split threshold				CRTS	
	Dis.	*T*	*RDP*(*Dis.*)%	*RDP*(*T*)%	*Dis.*	*T*	*RDP*(*Dis.*)%	*RDP*(*T*)%	*Dis.*	*T*
15	1,800.27	5.276	-3.18	-81.8	1,728.9	1.33	0.815	-27.82	1,742.99	0.96
20	182.712	5.15	-0.90	-81.8	173.99	1.35	4.067	-30.37	181.07	0.94
36	354.7	4.26	-1.25	-44.1	353.1	2.99	-0.805	-20.40	350.26	2.38



(a) (b)

Fig. 2 Computational results of the three instances in case study 2

4.3 Case study 3

To evaluate the CRTS performance on larger sized instances, we adopted six instances in case study 3 from CVRPLIB and compared the results obtained by SPLITABU in [4] and MRSILS in [13]. The basic information of customer number, total customer demand, capacity of vehicles, and vehicle number required in each instance is listed in Table 4.

In Table 5, the columns indicate the total travel distance (*Dis.*), consumed CPU time (*T*) by each of the three algorithms (the former two data come respectively from [4] and [13]), and *RDP* of the CRTS distance compared with those of the other two algorithms.

Table 4 Basic information in each instance in case study 3

Instance	Name	Customer number	Total demand	Capacity of vehicles	Vehicle number
3-1	vrpnc1	50	777	160	5
3-2	vrpnc2	75	1,364	140	10
3-3	vrpnc3	100	1,458	200	8
3-4	vrpnc4	150	2,235	200	12
3-5	vrpnc5	199	3,186	200	16
3-6	vrpnc11	120	1,375	200	7

Table 5 Computational results of the three algorithms

N	SPLITABU			MRSILS			CRTS	
	Dis.	*T*	*RDP*(*Dis.*)%	*Dis.*	*T*	*RDP*(*Dis.*)%	*Dis.*	*T*
50	527.66	17	5.69	531.03	24	5.02	557.71	2.18
75	853.61	64	7.91	831.85	24	10.74	921.16	5.94
100	840.12	60	12.61	834.52	24	13.37	946.06	6.26
150	1,055.08	440	14.45	1,066.04	24	13.28	1207.57	10.42
199	1,338.36	1,900	16.06	1,343.67	24	15.60	1553.32	13.38
120	1,056.96	39	2.04	1,048.00	24	2.91	1,078.47	8.34

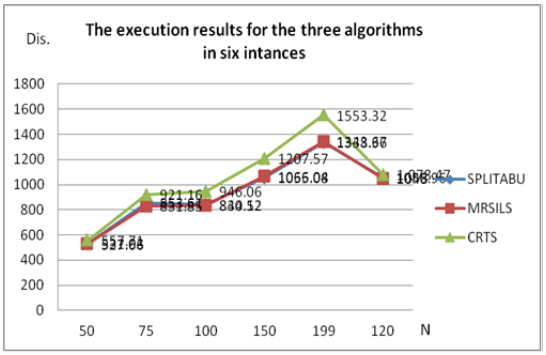


Fig. 3 Computational results for six instances in case study 3

Table 5 indicates the following. (1) The total travel distance of the CRTS is the highest among the three algorithms, and *RDPs* (*Dis.*) of SPLITABU and MRSILS are all positive. (2) When $N = 50$ and $N = 75$, the *RDP* (*Dis.*) values of SPLITABU and MRSILS are lower than or very close to 10 %. (3) When $N = 120$ in which all customer points are in a cluster distribution around the depot, the *RDP* (*Dis.*) values of SPLITABU and MRSILS are lower than 3 %. (4) The consumed CPU time by the CRTS is much lower than those by the other two algorithms.

Fig. 3 clearly shows that the CRTS is feasible and effective. Fig. 3 shows the curves of the total travel distances of the three algorithms in each of the three instances, namely, $N = 50$, $N = 75$, and $N = 120$. It also shows that the gap in the total travel distances between SPLITABU and the CRTS in each of the three instances, namely, $N = 50$, $N = 75$, and $N = 120$, is relatively small. The CRTS achieves near-optimal results even though no iteration is performed in the first two stages.

5. Conclusion

In this study, a mathematical model for SDVRP was described. On the basis of the CRTS, a three-stage algorithm was proposed. In the first stage, clustering was employed to partition the domain of customer points into sub-domains. In the second stage, 'push-out' and 'pull-in' operations were adopted to adjust the load demand in each sub-domain according to vehicle capacity Q. In the third stage, a proven TSA for TSP was used to optimise the total travel distance. Numerical experiments for the three cases were performed, and the computational results showed that the proposed CRTS algorithm provides feasible and effective solutions to the SDVRP in most parts of the test data set. The CRTS obtained approximate optimal solutions in 75 % of the instances. Three competing algorithms, namely, TSVI, *k*-means clustering method, and clustering method with a split threshold, were evaluated alongside the proposed method. These algorithms

were based on a 'routing after grouping' strategy. The CRTS performance in cases 1 and 2 was very close to that of the other three evaluated algorithms in finding the total travel distance. Comparison was also conducted for the computational results in case 3, which is a benchmark data set. When N = 50 and N = 75 in which the customer points were uniformly distributed around the depot, the *RDP* (*Dis.*) values were 5.69 % and 7.91 %, respectively. When N = 120, i.e. where the customer points were in a cluster distribution around the depot, *RDP* (*Dis*) was less than 3 %. Interestingly, the consumed CPU time by the proposed algorithm was much lower than that of any of the other algorithms evaluated in this study.

Acknowledgement

This work was financed by the 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, 141-149, doi: 10.1287/trsc.23.2.141.
[2] Archetti, C., Speranza, M.G. (2012). Vehicle routing problems with split deliveries, *International Transactions in Operational Research*, Vol. 19, No. 1-2, 3-22, doi: 10.1111/j.1475-3995.2011.00811.x.
[3] Dror, M., Laporte, G., Trudeau, P. (1994). Vehicle routing with split deliveries, *Discrete Applied Mathematics,* Vol. 50, No. 3, 239-254, doi: 10.1016/0166-218X(92)00172-I.
[4] Archetti, 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] 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.
[6] Archetti, C., Speranza, M.G., Savelsbergh, M.W.P. (2008). An optimization-based heuristic for the split delivery vehicle routing problem, *Transportation Science*, Vol. 42, No. 1, 22-31, doi: 10.1287/trsc.1070.0204.
[7] Gulczynski, D., Golden, B., Wasil, E. (2010). The split delivery vehicle routing problem with minimum delivery amount, *Transportation Research Part E: Logistics and Transportation Review,* Vol. 46, No. 5, 612-626, doi: 10.1016/j.tre.2009.12.007.
[8] 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.
[9] 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.
[10] Lu, X.C., Chen, Q.B., Zhang, Z.J. (2014). The electric vehicle routing optimizing algorithm and the charging stations' layout analysis in Beijing, *International Journal of Simulation Modelling*, Vol. 13, No. 1, 116-127, doi: 10.2507/IJSIMM13(1)CO4.
[11] 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. 37, No. 8, 1015-1019, doi: 10.3969/j.issn.1003-5060.2014.08.024.
[12] Zhu, X.N., Yan, R., Zhang, Q. (2015). A promoted hybrid heuristic algorithm for two-dimensional multi-depots vehicle routing problem, *International Journal of Simulation Modelling*, Vol. 14, No. 3, 499-510, doi: 10.2507/IJSIMM14(3)CO11.
[13] Wen, Z.Z. (2015). *Researches on iterated local search for the split delivery vehicle routing problem*, Beijing Transportation University, Beijing, P.R. China.
[14] Wu, D.Q., Dong, M., Li, H.Y., Li, F. (2016). Vehicle routing problem with time windows using multi-objective co-evolutionary approach, *International Journal of Simulation Modellin*g, Vol. 15, No. 4, 742-753, doi: 10.2507/IJSIMM15(4)CO19.
[15] Tang, M., Gong, D., Liu, S., Zhang, H. (2016). Applying multi-phase particle swarm optimization to solve bulk cargo port scheduling problem, *Advances in Production Engineering & Management*, Vol. 11, No. 4, 299-310, doi: 10.14743/apem2016.4.228.
[16] 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.
[17] Cao, Q.K., Yang, K.W., Ren, X.Y. (2017). Vehicle routing optimization with multiple fuzzy time windows based on improved wolf pack algorithm, *Advances in Production Engineering & Management*, Vol. 12, No. 4, 401-411, doi: 10.14743/apem017.4.267.
[18] Johanyák, Z.C. (2017). A modified particle swarm optimization algorithm for the optimization of a fuzzy classification subsystem in a series hybrid electric vehicle, *Tehnički Vjesnik – Technical Gazette*, Vol. 24, Supplement 2, 295-301, doi: 10.17559/TV-20151021202802.

[19] Wang, C.L., Li, S.W. (2018). Hybrid fruit fly optimization algorithm for solving multi-compartment vehicle routing problem in intelligent logistics, *Advances in Production Engineering & Management*, Vol. 13, No. 4, 466-478, doi: 10.14743/apem2018.4.304.
[20] 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.
[21] Lee, C.-G., Epelman, M.A, White III, C.C., Bozer, Y.A. (2006). A shortest path approach to the multiple-vehicle routing problem with split pick-ups, *Transportation Research B: Methodological*, Vol. 40, No. 4, 265-284, doi: 10.1016/j.trb.2004.11.004.
[22] 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.
[23] 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.
[24] 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.
[25] Luo, Z., Qin, H., Zhu, W., Lim, A. (2017). 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.
[26] 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.
[27] Min, J., Jin, C., Lu, L. (2018). A three-stage approach for split delivery vehicle routing problem solving, In: *Proceedings of 8th International Conference on Logistics, Informatics and Service Sciences* (LISS), Toronto, Canada, 1-6, doi: 10.1109/LISS.2018.8593226.
[28] Zhou, J., Xiong Z.Y., Zhang Y.F., Ren, F. (2006). Multiseed clustering algorithm based on max-min distance algorithm, *Journal of Computer Applications*, Vol. 26, No. 6, 1425-1427.