Split-Delivery Capacitated Arc-Routing Problem with Time Windows

Qidong Lai, Zizhen Zhang*, Mingzhu Yu, Jiahai Wang, *Senior Member, IEEE*

***Abstract*—Motivated by some practical applications in urban services such as water sparkling, we study a split-delivery capacitated arc-routing problem with time windows (SDCARPTW). It is a variant of arc-routing problem and is defined on an undirected graph where the demands on the arcs are splitable, and time window and capacity constraints must be satisfied. We propose a mathematical formulation for SDCARPTW and derive some nice properties of the split-delivery structure, which can help to well represent a solution of SDCARPTW. The dynamic programming, neighborhood search and perturbation process are combined to develop a tabu search algorithm. Through computational studies on CARPTW benchmark datasets, we validate the effectiveness and efficiency of our proposed algorithm. New datasets for SDCARPTW are further proposed and the impact of the split-delivery option is analyzed.**

I. Introduction

OVER the last few decades, we have witnessed the successful development of intelligent transportation systems (ITS). Today, in the cross-disciplinary fields between civil, mechanical, automation and operations research, new ITS technology are heavily involved with artificial intelligence and computational intelligence [1]. For example, with the increasing number of vehicles and activities of human being, ITS has been developed to relieve the traffic congestion [2]. In addition, the effective managements of vehicle scheduling and routing are also of importance in ITS [3]. In this paper, we mainly focuses on the routing optimization in ITS. In literature, most of the routing problems can be roughly classified into two main streams: vehicle routing and arc-routing. For vehicle routing, services are provided at the nodes of a network. For arc-routing, services are provided on the arcs (or edges). In this article, we focus on a variant of arc-routing problem (ARP), which is originated from some practical applications in urban services, such as garbage collection, snow removal, sweeping, gritting, mail delivery, school bus routing [4].

In practice, the services on a road network may be constrained by the vehicle capacity and time windows. For example, a sparkling vehicle has limited volume of its water tank. And it is not suitable to provide water sparkling services during traffic peak hours. The capacitated arc-routing problem with time windows (CARPTW) is therefore introduced to address these issues. In addition, it happens that the total demand on a certain arc may be higher than the vehicle capacity. In this case, multiple vehicles are necessary to serve a single arc demand. This is known as the *split-delivery* option. The split-delivery can also help to lower the overall traveling cost or the number of vehicles used.

In this paper, we initiate the study on the split-delivery capacitated arc-routing problem with time windows (SDCARPTW). SDCARPTW is generally more difficult to solve than CARPTW and SDCARP. Compared to CARPTW, additional decisions have to be made to determine the delivery quantity at each edge for each vehicle. Compared to SDCARP, the time window constraints may not be easy to respect. We derive some nice properties of SDCARPTW and then propose a tabu search algorithm to solve it. Through computational studies, we find that our algorithm can effectively deal with SDCARPTW instances compared with the existing approaches.

The remainder of the paper is organized as follows. Section II presents some related work on SDCARPTW. Section III formally describes the problem and builds a mathematical model. Section IV discusses the solution representations. Section V states how to evaluate a solution. Section VI gives a tabu search approach with its components. Section VII provides the computational results. Section VIII concludes our paper.

II. Literature Review

To the best of our knowledge, the only existing paper studied SDCARPTW was developed by Mullaseril et al. [5]. The authors modeled the feed delivery problem for the cattle ranch as a collection of capacitated rural postman problems with time-windows and split delivery, which is essentially SDCARPTW. They compared their solutions with the working practices on the cattle ranch.

SDCARPTW is a variant of the capacitated arc-routing problem (CARP), which is first investigated by Golden and Wong [6]. Eiselt et al. [7] [8] presented a detailed survey of CARP. Santos et al. [9] developed an ant colony optimization based metaheuristic with modifications in its components.

Manuscript received xx-xx-xxxx; revised xx-xx-xxxx; accepted xx-xx-xxxx. Date of publication xx-xx-xxxx; date of current version xx-xx-xxxx. This work is supported by the National Science Foundation of China (No. 71601191, 71771154) and Natural Science Foundation of Guangdong Province (No. 2019A1515011169). This paper is recommended by Associate Editor xxx.

Qidong Lai is with the Department of Management Sciences, City University of Hong Kong, Hong Kong S.A.R, China. (*E-mail*: qidong.Lai@my.cityu.edu.hk).

Zizhen Zhang is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, Guangdong, China. (*Corresponding author e-mail*: zhangzzh7@mail.sysu.edu.cn).

Mingzhu Yu is with the Institute of Big Data Intelligent Management and Decision, College of Management; College of Civil and Transportation Engineering, Shenzhen University, Shenzhen, 518060, China. (*E-mail*: mzyu@szu.edu.cn).

Jiahai Wang is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, Guangdong, China. (*E-mail*: wangjiah@mail.sysu.edu.cn).

Liu et al. [10] proposed a benchmark generator for CARP, which can control the size and complexity of road network resembling target application. Laporte et al. [11] solved a stochastic version of CARP. In addition to heuristic methods, exact methods such as branch-and-cut algorithm were used [12, 13]. Interestingly, Baldacci and Maniezzo [14] and Longo et al. [15] solved CARP by transforming it into the capacitated vehicle routing problem. They both mapped an arc of CARP into two nodes of CVRP in order to build a mathematical model.

SDCARPTW is also strongly related to CARPTW. Different methods have been proposed to solve CAPRTW. Ramdane-Cherif [16] was the first work dedicated to CARPTW. They presented a mathematical model and proposed a memetic algorithm with new memetic operators to tackle the time window constraints. A greedy randomized adaptive search procedure with path relinking was presented by Reghioui et al. [17]. Johnson and Wøhlk [18] proposed two column generation methods, as well as a heuristic method to obtain near optimal solutions of CARPTW. Ciancio et al. [19] defined a mixed capacitated general routing problem with time windows (MCGRPTW), where both arcs and nodes are associated with demands. They transformed MCGRPTW into an equivalent node routing problem over a directed graph with some restrictions.

There are also many papers working on the split-delivery capacitated vehicle routing problem (SDVRP) and their variants. Archetti et al. [20] applied a tabu search with operators to solve SDVRP. Chen et al. [21] reviewed applications of SDVRP and proposed a heuristic with mixed integer program and record-to-record travel algorithm. Archetti et al. [22] presented a branch-and-price-and-cut method for large cases of SDVRP. Archetti et al. [23], Desaulniers [24], Bianchessi and Irnich [25] tried to solve a variant of SDVRP called split delivery vehicle routing problem with time windows (SDVRPTW) by exact methods. Luo et al. [26] studied SDVRPTW and assumed that the traveling cost of an edge is a linear function of the vehicle load weight. Bianchessi et al. [27] took into account customer inconvenience constraints in SDVRPTW. Li et al. [28] focused on a variant of SDVRP, where there are more than one available time windows for each customer.

Some of the works believed that the split delivery could improve the solution quality in some scenarios. For example, the work by Dror and Trudeau [29] aims to demonstrate the potential of cost saving through split delivery. Mullaseril et al. [5] showed that, in four of five cases of current practice, the split delivery reduces the total distance traveled. Belenguer et al. [30] worked on SDCARP and thought that servicing an edge with a single vehicle is sometimes not realistic: vehicle capacity can be exhausted in the middle of a street segment or between two exists on a motor-way. Archetti et al. [31] pointed out that if all the customer demands are less than or equal to the vehicle capacity, it may be beneficial to use multiple vehicles to serve a customer.

III. Problem Statement and Model Building

SDCARPTW is defined on an undirected graph $G = (N, E)$, where $N = \{0, 1, ..., n\}$ is the node set and $E = \{(i, j) : i, j \in N, i \neq j\}$ is the edge set (here, edges have the same meaning as arcs). Node 0 is the depot. Each edge (i, j) is associated with a positive traveling cost $c_{i,j}$ and traveling time $t_{i,j}$. The corresponding matrices $[c_{i,j}]$ and $[t_{i,j}]$ satisfy the triangle inequality. If edge (i, j) is a service-required edge (SRE for short), it is further associated with an extra service time $s_{i,j}$. We assume that the incurred traveling time and the service time of an edge are proportional to its traveling cost with fixed ratios p_1 and p_2, respectively. If a vehicle passes a non-SRE (i, j) or an SRE (i, j) without serving it, only traveling time is incurred and $t_{i,j} = p_1 \cdot c_{i,j}$. Otherwise, the total required time is $t_{i,j} + s_{i,j} = p_2 \cdot c_{i,j}$.

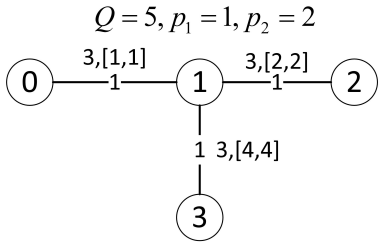

Fig. 1. A simple example of SDCARPTW.

We can classify the edges into m SREs and m' non-SREs. We mainly focus on SREs, since non-SREs only provide additional connectivity for the graph but impose no service constraints. For ease of descriptions, we also label the k-th SRE, say (i, j), as e_k (or $e(i, j)$), which has a demand d_k (or $d(i, j)$) and a service time window $[a_k, b_k]$ (or $[a(i, j), b(i, j)]$). The time window indicates that the service beginning time of e_k must be within the range $[a_k, b_k]$. Note that an edge (i, j) can be served in either direction, i.e., from i to j or from j to i.

There is a number of V vehicles initiated at the depot with an identical capacity Q. SDCARPTW aims to find V feasible routes such that the total traveling cost is minimized and all the demands are met. A route is denoted as a sequence of edges serviced by a vehicle starting from and ending at the depot. Suppose that $\delta_{i,k}$ is the quantity delivered to SRE e_k in route r_i. A route r_i is feasible if: 1) $\sum_{k=1}^{m} \delta_{i,k} \leq Q$, which means that the total service quantity of route r_i does not exceed Q; 2) the service beginning time of e_k included in route r_i respects the corresponding time window. Waiting at a node is possible when a vehicle arrives early. Because all the demands need to be satisfied, a feasible solution must have $\sum_{i=1}^{V} \delta_{i,k} = d_k, \forall k = 1, \ldots, m$.

Fig. 1 shows an example of SDCARPTW, which includes four nodes and three edges. Node 0 corresponds to the depot. Each edge is associated with three elements. The first one is the traveling cost given inside the edge (e.g., the traveling cost of $(0, 1)$ is 1). The other two are shown beside the edge, which give its demand and time window (e.g., the demand and time window of $(0, 1)$ are 3 and $[1, 1]$, respectively). The capacity is $Q = 5$ and the parameters are $p_1 = 1$ and $p_2 = 2$. For this example, let us consider a solution with two vehicles. The first vehicle starts from node 0 at time 1 and serves $(0, 1)$ 3 units. It arrives at node 1 at time 3. Then it visits $(1, 3)$ and serves 2 units. The second vehicle passes $(0, 1)$ and $(1, 2)$ without

service, and then serves $(1, 2)$ (from node 2 to node 1) 3 units at time 2. Then it serves $(1, 3)$ 1 units at time 4. Finally, two vehicles return to the depot. The total cost of this solution is 10. If we treat this example as CARPTW without the split-delivery option, at least three vehicles are required, thereby leading to greater traveling cost.

SDCARPTW is an NP-hard problem, as it can be easily proved by the reduction from SDCARP, which is NP-hard [30]. In the following, we try to formulate a mathematical model for SDCARPTW. It is built on a new directed graph G' transforming from graph G. Then the routing of arcs on G can be treated as the routing of nodes on G'. We first have the following theorem.

Theorem 1. *There always exists an optimal SDCARPTW solution in which an edge in a particular route is served by a vehicle at most twice (one service on each direction).*

Proof. If one direction of an edge e_k is served by a vehicle more than once, we can adjust this route by moving all the demands to the first service of this direction. Consequently, the cost of the route, the arrival time of the edges and the consumed capacity of the vehicle are not changed. Then we can delete all the service of this direction of e_k except the first one. The solution quality after the adjustment will not be impaired. □

In the original graph G, suppose that the k-th SRE e_k is connected with two ends i and j $(i < j)$. According to Theorem 1, each edge is served at most twice in a route. Then, e_k is transformed into two nodes in G' with their IDs k and $k + m$, which correspond to two directions of edge e_k. We use a pair of numbers u_k and v_k to describe node k in G' which indicates one direction of the edge in G, i.e., $(u_k, v_k) = (i, j)$ indicates servicing the edge from i to j. The node with ID $k + m$ relates to another direction of e_k, i.e., $(u_{k+m}, v_{k+m}) = (j, i)$. Two dummy nodes are also added in G' with IDs 0 and $2m + 1$. So G' consists of $2m + 2$ nodes in total. Such transformation ensures that a route in G has its counterpart route in G'. For instance, a route $0-1-3-1-0$ in G corresponds to a route $0-1-3-6-7$ in G'.

The transformation of edge attributes is as follows. Let $dist(i, j)$ be the shortest path between node i and node j in G. $c'_{i,j}$ is the cost of the edge between node i and node j in G', given by:

$$c'_{i,j} = \begin{cases} \frac{1}{2}[dist(u_i, v_i) + dist(u_j, v_j)] + dist(v_i, u_j), \\ \qquad \forall i, j \in [1, 2m] \\ \frac{1}{2}dist(u_j, v_j) + dist(0, u_j), \\ \qquad i = 0,\ \forall j \in [1, 2m] \\ \frac{1}{2}dist(u_i, v_i) + dist(v_i, 0), \\ \qquad \forall i \in [1, 2m],\ j = 2m + 1 \end{cases} \tag{1}$$

In graph G', the service time of node k and node $k + m$ $(k \in [1, m])$ is s'_k:

$$s'_k = s'_{k+m} = s_{u_k,v_k} \tag{2}$$

The traveling time of edge between node i and node j is $t'_{i,j}$, which can be obtained from $c'_{i,j}$ ($t'_{i,j} = p_2 \cdot c'_{i,j}$). The time window of the k-th ($i \in [1, 2m]$) node is $[a'_k, b'_k]$:

TABLE I
THE $dist$ MATRIX OF G.

dist	0	1	2	3
0	0	1	2	2
1	1	0	1	1
2	2	1	0	2
3	2	1	2	0

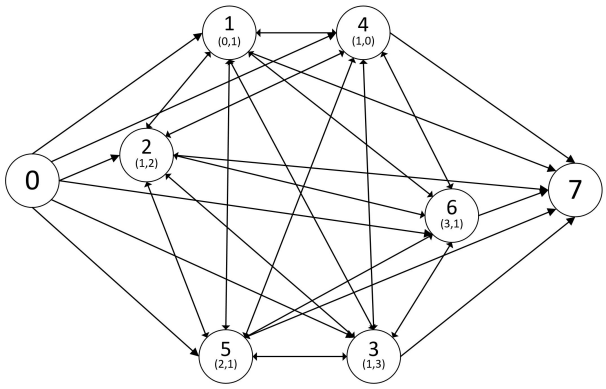

Fig. 2. A transformation of the first example.

$$[a'_k, b'_k] = \begin{cases} [a_k + t_{u_k,v_k}/2, b_k + t_{u_k,v_k}/2], \\ \qquad \forall k \in [1, m] \\ [a_{k-m} + t_{u_k,v_k}/2, b_{k-m} + t_{u_k,v_k}/2], \\ \qquad \forall k \in [m + 1, 2m] \end{cases} \tag{3}$$

Now we exemplify the transformation of the graph in Fig. 1. The $dist$ matrix is given in Table I. G' is shown in Fig. 2. The cost matrix of G' is presented in Table II. The time window and service time of a node in G' are depicted in Table III. It can be easily verified that such transformation ensures the equivalence of routing in G and in its counterpart G'.

TABLE II
THE COST MATRIX OF G'.

c'	0	1(0,1)	2(1,2)	3(1,3)	4(1,0)	5(2,1)	6(3,1)	7
0	-	0.5	1.5	1.5	1.5	2.5	2.5	-
1(0,1)	-	2	1	1	1	2	2	1.5
2(1,2)	-	3	2	2	2	1	3	2.5
3(1,3)	-	3	2	2	2	3	1	2.5
4(1,0)	-	1	2	2	2	3	3	0.5
5(2,1)	-	2	1	1	1	2	2	1.5
6(3,1)	-	2	1	1	1	2	2	1.5
7	-	-	-	-	-	-	-	-

TABLE III
THE TIME WINDOWS AND SERVICE TIME OF NODES IN G'.

	1(0,1)	2(1,2)	3(1,3)	4(1,0)	5(2,1)	6(3,1)
a'	1.5	2.5	4.5	1.5	2.5	4.5
b'	1.5	2.5	4.5	1.5	2.5	4.5
s'	1	1	1	1	1	1

Based on graph G', we can provide a mathematical formulation of SDCARPTW.

Decision variables:

$x_{i,j}^p$: a binary variable which is equal to 1 if vehicle p passes edge (i,j) in G', and 0 otherwise.

y_i^p: a binary variable which is equal to 1 if vehicle p serves node i in G', and 0 otherwise.

z_i^p: the quantity of node i served by vehicle p.

w_i^p: the arrival time when vehicle p reaches node i in G'.

$$minimize \quad \sum_{p=1}^{V} \sum_{i=0}^{2m+1} \sum_{j=0}^{2m+1} c'_{i,j} x_{i,j}^p \tag{4}$$

subject to:

$$\sum_{i=1}^{2m} x_{0,i}^p = 1, \quad \forall p \in [1,V] \tag{5}$$

$$\sum_{i=1}^{2m} x_{i,2m+1}^p = 1, \quad \forall p \in [1,V] \tag{6}$$

$$\sum_{j=0}^{2m} x_{j,i}^p - \sum_{j=1}^{2m+1} x_{i,j}^p = 0, \quad \forall i \in [1,2m],\ p \in [1,V] \tag{7}$$

$$y_i^p = \sum_{j=0}^{2m} x_{j,i}^p \le 1, \quad \forall p \in [1,V], \forall i \in [1,2m] \tag{8}$$

$$z_i^p \le y_i^p d_i, \quad z_{i+m}^p \le y_{i+m}^p d_i, \quad \forall p \in [1,V],\ \forall i \in [1,m] \tag{9}$$

$$\sum_{i=1}^{2m} z_i^p \le Q, \quad \forall p \in [1,V] \tag{10}$$

$$y_i^p a'_i \le w_i^p \le y_i^p b'_i + (1 - y_i^p) \cdot \infty, \quad \forall p \in [1,V], \forall i \in [1,2m] \tag{11}$$

$$w_i^p + s'_i x_{i,j}^p + t'_{i,j} - w_j^p \le (1 - x_{i,j}^p) \cdot \infty, \quad \forall i \in [0,2m], \forall j \in [1,2m+1] \tag{12}$$

$$x_{i,j}^p \in \{0,1\} \tag{13}$$

$$y_i^p \in \{0,1\} \tag{14}$$

$$z_i^p \in [0, d_i] \tag{15}$$

The objective (4) is to minimize the total traveling cost. Constraints (5) and (6) require that a vehicle must start from and return to the depot in each route, respectively. Constraints (7) ensure that the route continuity of the vehicle. Constraints (8) guarantee that each node in G' is served by a vehicle at most once. Constraints (9) make sure that the delivery quantity at a node cannot exceed its demand. Constraints (10) require that SREs are fully served. Constraints (11) mean that the sum of delivery quantity of a vehicle cannot exceed its capacity. Constraints (12) are the time window constraints. Constraints (13) ensure the connectivity of time between adjacent visiting.

It is worth noting that the mathematical model is very difficult to solve by using mixed-integer-linear-programming solvers, especially on large-size instances. Therefore, meta-heuristic approach may be more suitable to tackle the problem.

IV. Solution Representations

A solution can be represented by using three methods, i.e., sequence representation, flow network representation and forest representation. The first one indicates that a solution can be seen as V sequences of SREs with delivery quantities. For the second, we can view a solution as a flow network according to an assignment model. According to some analysis in our later discussions, the flow network can be simplified into a forest structure.

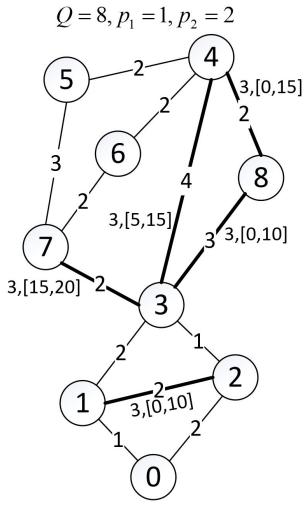

Fig. 3. A second SDCARPTW example.

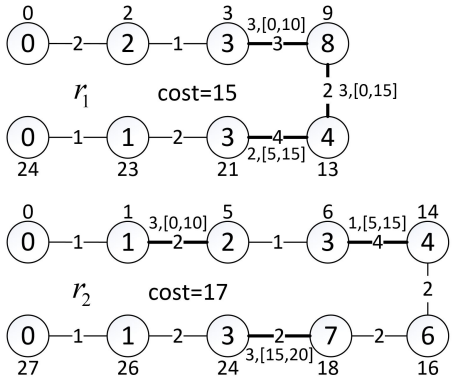

Fig. 4. Best solution of the example in Fig. 3.

In order to better explain these representations, another example of SDCARPTW is provided, as shown in Fig. 3. There are 9 nodes and 13 edges, in which 5 SREs are marked in bold. The format of elements associated with an SRE is the same as those in Fig. 1. For other non-SREs, the numbers inside them denote their traveling cost.

Fig. 4 shows the best solution of the example in Fig. 3. Two vehicles are dispatched to fulfill the total demand. The SREs are also marked in bold. The number inside an edge corresponds to its traveling cost. Two separated items are labelled on an SRE. The first one is the delivery quantity (not the demand) and the second one is the time window. Take edge $(3,7)$ in r_2 as an example, its traveling cost, delivery quantity and time window are 2, 3 and $[15,20]$, respectively. The arrival time of a node is marked beside it. For instance, the arrival time of node 7 in r_2 is 18.

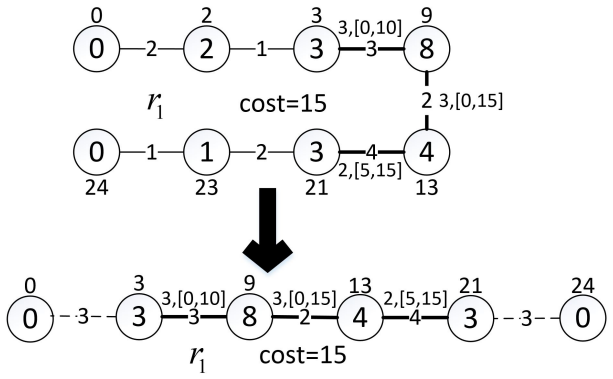

Fig. 5. A sequence of SREs.

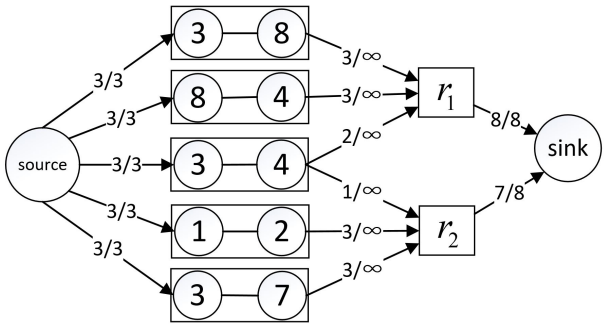

Fig. 6. A flow network $G(S)$.

A. Sequence Representation

We can simplify routes in a solution by only considering the SREs with their delivery quantities. By the shortest path property, continuous non-SREs in a route can be replaced by a shortest path between the first node and the last node in the non-SRE edge sequence. As shown in Fig. 5, the solid lines correspond to SREs and the dot lines represent shortest paths. Therefore, we can use $\langle(3,8),(8,4),(4,3)\rangle$ with their delivery quantities $\langle 3,2,4\rangle$ to represent r_1. The corresponding arrival time can be computed by using the shortest paths.

B. Flow Network Representation

A solution S consists of V routes. As discussed, each route can be represented by a sequence of SREs with delivery quantities. The delivery quantities can be further eliminated. If a sequence is given, we can determine a feasible delivery quantity on each SRE. This problem is essentially an assignment problem, which can be treated by using the maximum flow. The flow network, denoted as $G(S)$, is built as follows. Firstly, $G(S)$ contains a source node, a sink node, V route-nodes and m SRE-nodes. Secondly, add an edge between the source and each SRE-node $k(k=1,\ldots,m)$ with capacity d_k. Third, add an edge between each route-node $r(r=1,\ldots,V)$ and the sink with capacity Q. Finally, if route r serves SRE k in solution S, add an edge between route-node r and SRE-node k with an infinity capacity.

In this manner, the solution in Fig. 4 can be represented as the network in Fig. 6. If the maximum flow of the network is equal to $\sum_{k=1}^{m} d_k$ (i.e., all the SREs are fully served), there must exist a feasible assignment. Note that the complexity of maximum flow algorithms (e.g., Edmonds-Karp Algorithm) could be as high as $O((m+V)(mV)^2)$, where $m+V$ is the number of nodes and mV is the number of arcs in $G(S)$.

C. Forest Representation

We first provide the definition of a k-split cycle and a theorem as follows.

Definition (k-split cycle) Consider an SDCARPTW solution. Let an arbitrary k ($k>1$) SREs be $e_1, e_2, \ldots, e_k$, and k routes be $r_1, r_2, \ldots, r_k$, where r_i serves e_i and e_{i+1}, for $i=1,\ldots,k-1$ and r_k serves e_k and e_1. These subsets of edges and routes form a k-split cycle.

Fig. 7 shows an example of a 2-split cycle. The number inside an edge is the delivery quantity. Fig. 8 eliminates the cycle after some service adjustments.

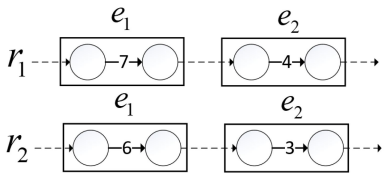

Fig. 7. An example of a 2-split cycle.

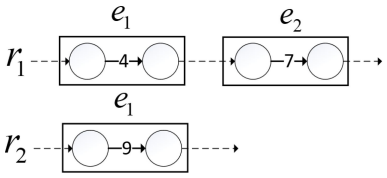

Fig. 8. Eliminating the cycle after some service adjustments.

Theorem 2. *There always exists an optimal solution for SDCARPTW that contains no k-split cycle ($k \geq 2$).*

Proof. [30] had proven that the theorem holds for SDCARP. The k-split cycle can be eliminated by some adjustments of service delivery quantities. When extending to SDCARPTW, such cycle elimination (i.e., deleting some edges from the cycle) would not break the time window constraints, since the vehicle can wait at a node to start services following the original time of the optimal solution before cycle elimination. □

The above discussions indicate that we can restrict our search space into those solutions without any k-split cycles. A solution S in such search space exhibits a forest structure (no cycles). Let us consider a graph $G'(S)$ which is built by removing the source node and the sink node from $G(S)$, and then transforming into undirected graph. Then $G'(S)$ is a forest. Fig. 9 shows a forest $G'(S)$ corresponding to Fig. 6.

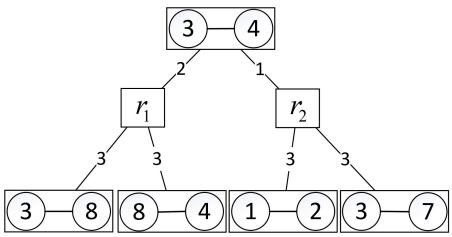

Fig. 9. A forest $G'(S)$ corresponding to Fig. 6.

V. Solution Evaluation

A. Objective Calculation

The objective value of a solution is the sum of the traveling cost of each route. For a particular route r, we assume that its SRE sequence is $\langle(u_1, v_1), (u_2, v_2), \ldots, (u_R, v_R)\rangle$, where R is the number of SREs in the route. It is worth noting that all the SREs are undirected and a vehicle can visit an edge in either directions. Therefore, there exists 2^R combinations of possible routing ways for an SRE sequence if the time window constraints are not considered. Fig. 10 shows possible routing ways of r_1 in Fig. 5. The vehicle starts from node 0, visits three SREs and goes back to node 0. In this process, the choices of routing are shown in dot lines and the number inside a line is the associated cost.

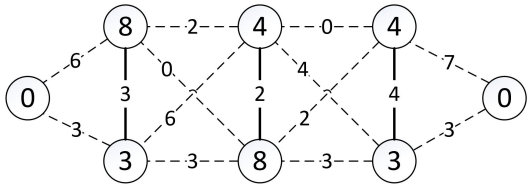

Fig. 10. Possible routing ways of a route.

We devise a dynamic programming approach to find the optimal routing way. Denote $MC[i][j]$ as the minimum traveling cost of the vehicle visiting the SRE sequence $\langle(u_1, v_1), (u_2, v_2), \ldots, (u_i, v_i)\rangle$, and finally arriving at node j (j must be either u_i or v_i). The state transition formulas are given as follows.

$$MC[i][u_i] = \min\{MC[i-1][u_{i-1}] + dist[u_{i-1}][v_i], MC[i-1][v_{i-1}] + dist[v_{i-1}][v_i]\} + c(v_i, u_i) \quad (16)$$

$$MC[i][v_i] = \min\{MC[i-1][v_{i-1}] + dist[v_{i-1}][u_i], MC[i-1][u_{i-1}] + dist[u_{i-1}][u_i]\} + c(u_i, v_i) \quad (17)$$

The boundary conditions are:

$$MC[1][u_1] = dist[0][v_1] + c(v_1, u_1) \quad (18)$$

$$MC[1][v_1] = dist[0][u_1] + c(u_1, v_1) \quad (19)$$

The optimal traveling cost of the given SRE sequence is:

$$\min\{MC[R][u_R] + dist[u_R][0], MC[R][v_R] + dist[v_R][0]\} \quad (20)$$

For SDCARPTW, we further need to consider the time window constraints for these 2^R combinations. We could find those combinations respecting the time windows. Observe that a vehicle could start its service within the time window as soon as it arrives at a node. We thereby use another dynamic programming to calculate the earliest service beginning at two end nodes of each SRE in route r. Let $ET[i][j]$ denote such time at node j in the i-th SRE given the SRE sequence $\langle(u_1, v_1), (u_2, v_2), \ldots, (u_i, v_i)\rangle$. It can be recursively calculated as follows.

$$ET[i][u_i] = \max\{\min\{ET[i-1][u_{i-1}] + p_2 c(u_{i-1}, v_{i-1}) + p_1 dist[v_{i-1}][u_i], ET[i-1][v_{i-1}] + p_2 c(v_{i-1}, u_{i-1}) + p_1 dist[u_{i-1}][u_i]\}, a(u_i, v_i)\} \quad (21)$$

$$ET[i][v_i] = \max\{\min\{ET[i-1][u_{i-1}] + p_2 c(u_{i-1}, v_{i-1}) + p_1 dist[v_{i-1}][v_i], ET[i-1][v_{i-1}] + p_2 c(v_{i-1}, u_{i-1}) + p_1 dist[u_{i-1}][v_i]\}, a(u_i, v_i)\} \quad (22)$$

In Equation (21), two separate terms inside the function $\min$ determines the earliest arrival time reaching node u_i: one goes from u_{i-1} and the other goes from v_{i-1}. Note that the coefficients p_1 and p_2 help to convert the cost into time. Because the vehicle cannot start service edge (u_i, v_i) before its ready time $a(u_i, v_i)$, the function $\max$ is introduced to get a proper $ET[i][u_i]$. The explanation of Equation (22) is similar to that of Equation (21).

The boundary conditions are:

$$ET[1][u_1] = \max\{p_1 dist[0][u_1], a(u_1, v_1)\} \quad (23)$$

$$ET[1][v_1] = \max\{p_1 dist[0][v_1], a(u_1, v_1)\} \quad (24)$$

Note that when the time window constraints are considered, the update of $MC[i][j]$ must depend on $ET[i][j]$. For example, if $ET[i][u_i] > b(u_i, v_i)$, the service of (u_i, v_i) becomes impossible. In this case, $MC[i][v_i]$ is undefined or set to infinity. Similarly, if $ET[i][v_i] > b(u_i, v_i)$, then $MC[i][u_i]$ is undefined. In our solution approach, we choose to keep or relax the time window constraints for the objective calculation during the search process. Detailed discussions can be found in Section VI.

The time complexity of each dynamic programming approach is $O(R)$, because there exist $2R$ states and the state transition only takes $O(1)$. Compared with a simple enumeration method, the dynamic programming approach is much more efficient.

B. Feasibility Checking

In Section IV, we mentioned that the sequence representation needs to take into account the delivery quantity at each SRE. For the flow network or forest representation, we need to find a feasible delivery quantity assignment. Because the flow network representation requires a great deal of time to find a maximum flow, the forest representation is more desirable. In the literature, [32] proposed a greedy algorithm to check the feasibility of the forest structure for SDVRP. This method can

also be adapted to deal with SDCARPTW, which is briefly stated as follows.

A forest contains a set of unrooted trees. For each unrooted tree, we need to check whether all the SRE-nodes in the tree can be fully served. We arbitrarily select a route-node as the root node, creating a rooted tree. Denoted $residual(r)$ as the residual capacity of route-node r, and $residual(k)$ as the residual demand of SRE-node k. Initially, set $residual(r) = Q$ for each route-node r and set $residual(k) = d_k$ for each SRE-node k. Then, we try to use depth first search or breadth first search to examine all the tree nodes from leaves to the root. If the current examining node is an SRE-node (denoted as r), then its father must be a route-node (denoted as k). The delivery quantity $\delta_{r,k}$ is calculated by:

$$\delta_{r,k} = residual(r) \tag{25}$$

And the residuals are updated by

$$residual(k) \leftarrow residual(k) - \delta_{r,k} \tag{26}$$

$$residual(r) \leftarrow 0 \tag{27}$$

If the current examining node is a route-node (also denoted as k), then its father is an SRE-node (denoted as r). In this case, $\delta_{r,k}$ and residuals are given by:

$$\delta_{r,k} = \min\{residual(r), residual(k)\} \tag{28}$$

$$residual(k) \leftarrow residual(k) - \delta_{r,k} \tag{29}$$

$$residual(r) \leftarrow residual(r) - \delta_{r,k} \tag{30}$$

The unrooted tree is infeasible if a route node r encounters a negative $residual(r)$; otherwise, the tree must have a feasible assignment.

As an example, we present the detailed steps of processing a tree as shown in Fig. 9 based on the example shown in Fig. 6. 1) SRE-nodes $(3, 8)$ and $(4, 8)$ are fully served by route-node r_1. $(1, 2)$ and $(3, 7)$ are fully served by route-node r_2. The residual capacities of r_1 and r_2 are all equal to $8-3-3 = 2$. 2) r_1 delivers 2 unit quantities to SRE-node $(3, 4)$. Similarly, r_2 delivers 1 unit quantity to SRE-node $(3, 4)$. 3) SRE-node $(3, 4)$ is the root node and the greedy algorithm terminates with a feasible assignment of delivery quantities.

The complexity of the greedy algorithm only takes $O(m + V)$. Therefore, it is quite efficient compared with general network flow algorithms.

VI. Tabu Search Algorithm

In this section, we propose a standard tabu search framework for solving SDCARPTW. Algorithm 1 presents the pseudocode. It contains an initialization phase, a neighborhood search phase and a perturbation phase. In the neighborhood search, we allow that a solution violates the time window constraints. In other words, we treat them as soft constraints.

A. Initial Solution

An initial solution of SDCARPTW is built by using a simple construction method described as follows.

Step 1: Sort all SREs in ascending order of the latest available service time (i.e., b_i).

Algorithm 1 Framework of the tabu search algorithm for SDCARPTW.

```
S ← an initial solution S_0
S_best ← S
Initialize the tabu list
while MaxPertubation is not reached do
    while MaxNonImprovement is not reached do
        Find the best solution S' in the neighborhood of S;
        S ← S'
        Update tabu list
        if S' respects the time window constraints then
            Update S_best with S'
        end if
    end while
    S ← Perform perturbation on solution S_best
end while
return S_best
```

Step 2: Select an SRE e_i which has not been satisfied.

Step 3: Check if there exists a route r that can accommodate e_i without violating the time window constraints. If none, go to Step 4, otherwise go to Step 5.

Step 4: Open an empty route with capacity Q. Go to Step 3.

Step 5: Insert e_i into the best feasible position of r (with the minimum insertion cost). The following two cases are happened. 1) If the residual capacity of r can satisfy the demand d_i, e_i is fully served by r and the residual capacity of r is subtracted by d_i. 2) If the residual capacity of r is smaller than the demand d_i, e_i is partially served by r using all its residual capacity. Then d_i is updated.

Step 6: Go to Step 2 until all the SREs have been satisfied.

It is guaranteed that the construction method terminates with a feasible solution constructed and does not generate any k-split cycles.

B. Time Window Relaxation

For some cases, it is difficult to find a feasible neighboring solution due to the tightness of time windows. If we allow the occurrence of an infeasible solution, we may achieve more promising feasible solutions in future steps. To this end, we relax the time window constraints in the neighborhood search by introducing a penalty function $p(S)$ to measure the degree of violations of time windows. In particular, if S is feasible, then $p(S) = 0$. Otherwise, $p(S)$ is empirically set as:

$$p(S) = p_2 \cdot AC \cdot W(S) \tag{31}$$

where AC is the average cost of all the edges, $W(S)$ is the total number of SREs violating the time window constraints in solution S.

The fitness of a solution S is its total traveling cost plus the penalty function $p(S)$. In the neighborhood search phase, we use the fitness to guide the search direction. Note that the best feasible solution is recorded during the neighborhood search. We should guarantee that the final returned solution satisfies the time window constraints.

C. Neighborhood Operators

We use four forest-based operators to define the neighborhood of a solution. Most of these operators are similar to the ones proposed in Zhang et al. [32]. Note that the time window constraints are excluded in these operators.

1) Relocation Operator: Given a solution S, the relocation operator tries to move an SRE $e_i (i \in [1, m])$ from route $r_k (k \in [1, V])$ serving it, to another route $r_l (l \in [1, V]$, $l \neq k$ and $e_i \notin r_l)$. After the relocation, the forest structure must be kept. The procedure of this operator is follows. First, cut the edge (e_i, r_k) in the forest. Second, link SRE-node e_i to route-node r_l.

2) Exchange Operator: The exchange operator tries to exchange an SRE $e_i (i \in [1, m])$ from its serving route $r_k (k \in [1, V])$ with another SRE $e_j (j \in [1, m]$, $i \neq j$ and $e_j \notin r_k)$ from its serving route $r_l (l \in [1, V]$, $l \neq k$ and $e_i \notin r_l)$. After the exchange, the forest structure must be guaranteed.

3) Swap(1,1) Operator: This operator is an extension of exchange operator which is originally proposed by [33]. When the exchange operator produces an infeasible result, the swap(1,1) is invoked to repair it.

4) Split Operator: The above three operators have little chances to split demand into multiple routes. The split operator is therefore devised to cater to the need of demand splitting. It first randomly selects an SRE e_i. Next, e_i is removed from all the routes serving it. Last, reinsert e_i into the best position of existing routes until the demand of e_i is satisfied. Because the reinsertion needs to examine every possible positions and is time-consuming, the split operator is performed once per μ iterations.

D. Perturbation

The perturbation process is invoked when a solution is possibly trapped into a local optima. It could be viewed as a *multi-split* operator, i.e., a set of SREs are removed from solution S and then reinserted into S by using the *split* operator. Detailed process is provided as follows.

Step 1: Randomly generate a value λ from the user-defined range $[\lambda_1, \lambda_2]$.

Step 2: Randomly select a set T of SREs, where $|T| = \lambda$.

Step 3: Remove all the SREs of T from S and get S'.

Step 4: Insert all the SREs of T into S' by using the *split* operator, resulting in a perturbed solution S.

E. Tabu List

To prevent the search from going back to previously visited solutions, we use a tabu list to record some potentially "bad" operations. If an SRE e_i is moved from r_k to r_l by relocation operator or exchange operator, e_i is forbidden to be inserted into r_k or removed from r_l in the next β_1 iterations. If an SRE e_i is split, it is not allowed to split in the next β_2 iterations. Here, β_1 and β_2 are user-define parameters.

VII. Computational Experiments

The proposed tabu search algorithm was coded in C++. All the experiments were conducted on a computer with an Intel(R) Core(TM) i5-4590 CPU @ 3.3GHz, 8G RAM and windows 10 operation system.

A. Datasets and Parameters

There are no benchmark SDCARPTW datasets available in the existing studies. However, we have found four CARPTW datasets proposed by Wøhlk [34] and each of them contains 8 instances with different scales. These datasets mainly differ in the interval of their time windows. More detailed descriptions of the datasets can be found in Reghioui et al. [17]. For a particular CARPTW instance, it contains all the information as indicated in Section III, so it can be directly used for SDCARPTW. It is worth noting that Ciancio et al. [19] also proposed some CARPTW instances, in which the time windows are considered for all the edges but not only SREs. Therefore, it is not suitable to apply our algorithm to tackle these instances.

In our algorithm, there are 7 parameters in total. Their values were carefully calibrated and the final settings are listed in Table IV.

TABLE IV
Parameter settings.

Parameter	Definition	Value
$MaxPertubation$	The maximum number of perturbation time	10
$MaxNonImprovements$	The maximum non-improvement iterations	3000
μ	The period for calling the *split* operator	200
$[\lambda_1, \lambda_2]$	The range of value for removing SREs in perturbation	[3,7]
$\{\beta_1, \beta_2\}$	Tabu tenures	$\{2\sqrt{m}, 3.5m\}$

B. Computational Results

The results on four sets of CARPTW instances are presented in Table V. The first column gives the names of instances. The second column indicates the size of the edge set. The third, fourth and fifth columns are the results provided by CARPTW solvers which can be found in the experiment part of Reghioui et al. [17], where the heading "LB" is the lower bound obtained by Wøhlk [34]. "PNH" is a Preferable Neighbor Heuristic proposed by Wøhlk [34]. "GRASP" gives the best result obtained by Reghioui et al. [17]. The performance of our tabu search algorithm is shown in the last three columns. They record the minimum cost, average cost over 10 runs and the average computation time (in seconds). To our best knowledge, there is no SDCARPTW algorithm in the existing literature.

In our solution results, the value marked in bold is as good as the optimal solution obtained by CARPTW solver, while the value marked with an asterisk is better than the solution of CARPTW. It is worth pointing out that the CARPTW solution on B20B is optimal (with the value 214). However, if we treat

TABLE V
THE RESULTS ON CARPTW DATASETS.

Instance	$\|E\|$	CARPTW algorithms			Our approach		
		LB	PNH	GRASP	Best	Average	Time
A10A	15	107	107	107	**107**	107	1.5
A13A	23	202	202	202	**202**	202	5.0
A13B	23	171	173	171	**171**	171	5.7
A13C	23	163	163	163	**163**	163	5.8
A20B	31	260	264	260	**260**	260	9.1
A40C	69	660	660	660	**660**	660	99.0
A40D	69	807	807	807	809	809.8	93.3
A60A	90	1822	1822	1830	1829	1841.2	196.1
B10A	15	87	87	87	**87**	87	0.9
B13A	23	167	167	167	**167**	167	4.6
B13B	23	152	158	152	**152**	152	4.8
B13C	23	141	141	141	**141**	141	5.9
B20B	31	214	214	214	210*	210	18.2
B40C	69	588	602	602	**602**	602	106.3
B40D	69	730	730	730	**730**	733.1	112.7
B60A	90	1554	1554	1565	1559	1574.2	269.6
C10A	15	73	73	73	**73**	73	3.0
C13A	23	142	148	142	**142**	142.7	6.3
C13B	23	132	132	132	**132**	132	6.1
C13C	23	121	121	121	**121**	121	5.8
C20B	31	186	186	186	188	191	23.2
C40C	69	503	563	547	544*	544.9	122.9
C40D	69	611	626	632	627	630.8	121.2
C60A	90	1283	1283	1300	1300	1325.7	336.9
D10A	15	-	-	-	87	87	1.0
D13A	23	-	-	-	167	167	4.6
D13B	23	-	-	-	152	152	5.5
D13C	23	-	-	-	141	141	6.3
D20B	31	-	-	-	210	210	11.6
D40C	69	-	-	-	602	602.4	111.6
D40D	69	-	-	-	730	732	112.4
D60A	90	-	-	-	1545	1557.4	251.6

the instance as SDCARPTW, the cost can be even lower (with the value 210).

We also tried to use the commercial solver CPLEX to solve the SDCARPTW model given in Section III. We find that only the instances A10A, B10A and C10A can be optimally solved within 1 hour. The corresponding optimal results are 107, 87 and 73, respectively. For other instances, our algorithm outperforms CPLEX in terms of both running time and solution quality.

C. Analysis of Split Option

The results in Table V show that our approach cannot improve CARPTW solvers in a great extent. This may be due to the fact that the split option has very little effect on these instances. We thereby analyze how the structure of an instance relates to the split rate in the final solution. Here, the split rate of a solution is calculated by the total number of SREs appeared in the solution (note that an SRE may be appeared multiple times in different routes) over m (the number of SREs). The larger value of the split rate indicates more split services.

Table VI lists the indicators for the analysis of split options, as well as their definitions and formulas. The column "Factor Group" is used for analyzing the reasons, which will be detailed in the following discussions. Table VII presents the values of these indicators on all the instances. The row marked in bold means that the corresponding split rate is greater than 1. As we can see, for most instances, the split rate is equal to 1. We believe that this is due to the following reasons.

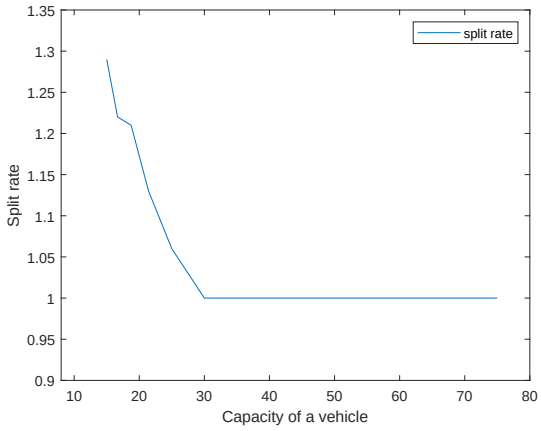

Fig. 11. Sensitivity analysis of the split rate.

1) The Interval of the Time Window is Tight: RAM is the ratio of $ALTW$ to $MLTW$. A small RAM implies that the corresponding interval of time window is tight. From Table VII, we find that the RAM values on all the instances except B10A, C10A, C20B, C60A and D10A are very small. This means that a vehicle can only start service for an edge in a very short time period. In this case, there is no incentive for an edge to use split service, since it would be difficult for multiple vehicles to perform service within the time window.

2) The Vehicle Capacity is Large: The indicator $ENSE$ is the expected number of served edges per vehicle, which is calculated by ACT over ATE. It assumes that a vehicle continuously served edges within the allowed time. The indicator ECC is the expected consumed the capacity of a vehicle, which is calculated by the product of average demand of edges (TD/m) and $ENSE$. Compared ECC with Q, we know the utilization of vehicle capacity. Observed from Table VII, ECC is only greater than Q on the instances B60A, C10A, C20B and C60A. For other instances, ECC is smaller than Q. It is not likely that the split service would happen if the utilization of vehicle capacity is low, as one vehicle suffices to fulfil the edge demand.

We further conducted an experiment on a selected instance B40D to analyze the sensitivity of split rate with the vehicle capacity. The initial vehicle capacity is 75. We tried to reduce it gradually. As we can see from Fig. 11, when the vehicle capacity becomes small, the split rate grows from 1 to around 1.3. This implies that the split option becomes significant.

D. New Datasets

According to the above analysis, the CARPTW datasets are not suitable for SDCARPTW. Thus, we propose new SDCARPTW datasets by modifying the time windows and the demand of SREs of CARPTW instances. Due to the tight interval of time windows in original instances, we enlarge the them in new instances. If the original time window of the i-th SRE is

TABLE VI
INDICATORS FOR THE ANALYSIS OF SPLIT OPTIONS.

Factor Group	Indicator	Definition	Formula
-	SR	Split rate	$(No.\ SREs\ appeared\ in\ the\ solution)/m$
-	TD	The total demand of edges	$\sum_{i=1}^{m} d_i$
-	ATE	The average service time of edges	$\sum_{i=1}^{m} p_2 c_i/m$
-	ACT	The average completion time	$\sum_{i=1}^{m} b_i/m$
-	$ENSE$	The expected number of served edges per vehicle	ACT/ATE
1	$ALTW$	The average length of time windows	$\sum_{i=1}^{m}(b_i - a_i)/m$
1	$MLTW$	The maximum latest time of time windows	$max\{b_i \mid i \in [1, m]\}$
2	RAM	The ratio of $ALTW$ to $MLTW$	$ALTW/MLTW$
2	ECC	The expected consumed capacity of a vehicle	$TD/m \cdot ENSE$
2	UV	The utilization of a vehicle	ECC/Q

TABLE VII
THE VALUES OF DIFFERENT INDICATORS.

Instance	SR	TD	ATE	$ALST$	$ENSE$	Factor Group 1			Factor Group 2		
						$ALTW$	$MLTW$	RAM	ECC	Q	UV
A10A	1	53	7.82	24.91	3	9.45	31	0.30	15.35	25	0.61
A13A	1	98	19.32	106.77	5	17.82	132	0.13	24.62	50	0.49
A13B	1	98	15.45	106.77	6	17.82	132	0.13	30.78	50	0.62
A13C	1	98	11.59	106.77	9	17.82	132	0.13	41.03	50	0.82
A20B	1	152	11.69	45.41	3	12.21	79	0.15	20.36	25	0.81
A40C	1	325	15.67	153.06	9	22.41	222	0.10	50.40	75	0.67
A40D	1	325	26.11	153.06	5	22.41	222	0.10	30.24	75	0.40
A60A	1	423	13.48	56.98	4	15.51	70	0.22	22.07	25	0.88
B10A	1	53	7.82	34.91	4	19.45	41	0.47	21.51	25	0.86
B13A	1	98	19.32	116.77	6	27.82	142	0.20	26.93	50	0.54
B13B	1	98	15.45	116.77	7	27.82	142	0.20	33.66	50	0.67
B13C	1	98	11.59	116.77	10	27.82	142	0.20	44.88	50	0.90
B20B	**1.07**	**152**	**11.69**	**55.41**	**4**	**22.21**	**89**	**0.25**	**24.85**	**25**	**0.99**
B40C	1	325	15.67	163.06	10	32.41	232	0.14	53.69	75	0.72
B40D	1	325	26.11	163.06	6	32.41	232	0.14	32.22	75	0.43
B60A	1	423	13.48	67.35	4	25.88	80	0.32	26.09	25	1.04
C10A	**1.09**	**53**	**7.82**	**44.00**	**5**	**38.55**	**51**	**0.76**	**27.12**	**25**	**1.08**
C13A	1	98	19.32	126.77	6	47.82	152	0.31	29.23	50	0.58
C13B	1	98	15.45	126.77	8	46.91	152	0.31	36.54	50	0.73
C13C	1	98	11.59	126.77	10	47.82	152	0.31	48.72	50	0.97
C20B	**1.1**	**152**	**11.69**	**65.41**	**5**	**41.48**	**99**	**0.42**	**29.33**	**25**	**1.17**
C40C	1	325	15.67	172.56	11	52.98	242	0.22	56.82	75	0.76
C40D	1	325	26.11	173.06	6	52.41	242	0.22	34.19	75	0.46
C60A	**1.01**	**423**	**13.48**	**77.35**	**5**	**46.00**	**90**	**0.51**	**29.96**	**25**	**1.20**
D10A	1	53	7.82	34.91	4	19.45	41	0.47	21.51	150	0.14
D13A	1	98	19.32	116.77	6	27.82	142	0.20	26.93	150	0.18
D13B	1	98	15.45	116.77	7	27.82	142	0.20	33.66	150	0.22
D13C	1	98	11.59	116.77	10	27.82	142	0.20	44.88	150	0.30
D20B	1	152	11.69	55.41	4	22.21	89	0.25	24.85	150	0.17
D40C	1	325	15.67	163.06	10	32.41	232	0.14	53.69	150	0.36
D40D	1	325	26.11	163.06	6	32.41	232	0.14	32.22	150	0.21
D60A	1	423	13.48	67.35	4	25.88	80	0.32	26.09	150	0.17

$[a_i, b_i]$, the new one is set to $[max(0, \frac{3a_i - b_i}{2}), \frac{3b_i - a_i + 2}{2}]$. Due to the large capacity of the vehicle, we enlarge the demand of SREs by multiplying 5 on set A, B, C instances, and 8 on set D instances.

The results on the new SDCARPTW instances are shown in Table VIII. We also use a CARPTW algorithm analogous to Wøhlk [34] to solve the instances. The columns "NV", "Cost" and "SR" give the number of vehicles used, total traveling cost and split rate, respectively. On average, the split delivery can reduce 9% of "NV" and 14% of "Cost". For those instances in the lower part of the table, they have no feasible solutions for CARPTW, since some SREs have exceeded the vehicle capacity. The split rates SR for all the instances are greater than 1, which indicate that the split-delivery option is applied significantly.

To further test the convergence of our algorithm and the effects of different operators, we arbitrarily selected one instance in new dataset (B40D) for the experiments. When an algorithm was running, we recorded the currently best solution per every second. The results are plotted in Fig. 12. The blue line corresponds to the solutions of our algorithm (called "full operators"). Other three lines represent the solutions of the other three versions, in which the relocation operator, exchange operator, split operator are respectively removed. Because different operators have different complexities, the running time of these three versions is different. Therefore,

we set the termination of all the algorithms to the same time by properly adjusting the parameter $MaxPerturbation$.

The figure shows that all the versions can greatly improve the initial solutions in the early stage. The green line corresponds to the algorithm without the relocation operator. We can see that the final solution obtained by it is the worst. This means that the relocation operator plays the most important role in the neighborhood search phase. Both the black line and red line indicate that the corresponding versions have very fast convergence in the first 50 seconds. However, they are dominated by the "full operators" version in later iterations. It seems that the split operator is slightly more useful than the exchange operator. The blue line shows the convergence of the proposed algorithm ("full operators"). It can attain the most desirable solution compared with the other versions. The fact partly verifies the effectiveness of the proposed operators.

Fig. 12. The convergence plot of different versions of algorithms on the instance B40D.

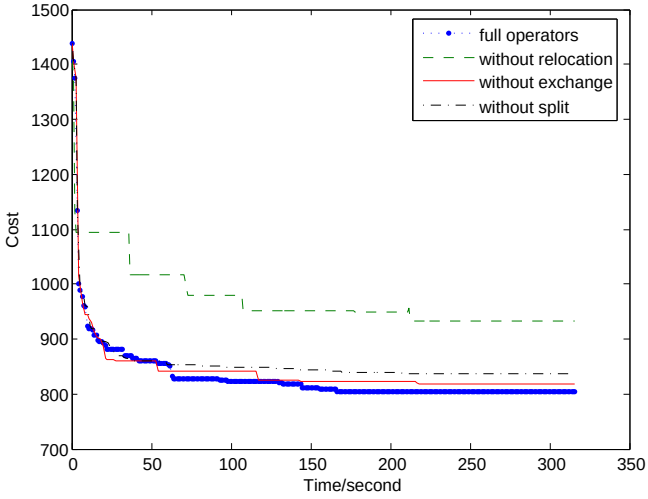

TABLE VIII
THE RESULTS ON NEW SDCARPTW DATASETS.

Instance	CARPTW		SDCARPTW		
	NV	Cost	NV	Cost	SR
SD-A13A	11	228	10	211	1.41
SD-A13B	12	237	10	210	1.41
SD-A13C	11	231	10	210	1.36
SD-A40C	23	921	22	797	1.3
SD-A40D	24	909	22	816	1.27
SD-B13A	11	249	10	210	1.36
SD-B13B	12	233	10	210	1.36
SD-B13C	12	233	10	210	1.32
SD-B40C	23	940	22	792	1.32
SD-B40D	23	898	22	799	1.33
SD-C13A	12	233	10	210	1.36
SD-C13B	12	233	10	210	1.36
SD-C13C	12	233	10	210	1.32
SD-C40C	23	913	22	795	1.33
SD-C40D	23	923	22	794	1.32
SD-D10A	3	93	3	75	1.18
SD-D13A	6	154	6	143	1.14
SD-D13B	6	168	6	143	1.14
SD-D13C	6	163	6	143	1.18
SD-D20B	9	263	9	220	1.24
SD-D40C	19	833	18	688	1.25
SD-D40D	20	816	18	701	1.25
SD-D60A	26	1682	23	1488	1.26
Average	14.7	512.4	13.5	447.2	1.29
SD-A10A	-	-	11	189	1.64
SD-A20B	-	-	31	639	1.69
SD-A60A	-	-	85	4696	1.94
SD-B10A	-	-	11	189	1.64
SD-B20B	-	-	31	643	1.9
SD-B60A	-	-	85	4715	1.9
SD-C10A	-	-	11	189	1.73
SD-C20B	-	-	31	641	1.83
SD-C60A	-	-	85	4714	1.94

VIII. CONCLUSIONS

In this paper, we consider a capacitated arc-routing problem with time window constraints and splitable service quantity. It can deal with those cases in which multiple services are performed on the arcs/edges. We propose a mathematical formulation for SDCARPTW based on a graph transformation. Then we show how to represent and evaluate a solution. We develop an effective tabu search algorithm, which makes use of the forest structure of a solution. We conduct experiments on the benchmark datasets of CARPTW and analyze the characteristics of the results. We also generate new datasets which are more suitable for SDCARPTW. From the numerical studies, the efficiency and effectiveness of our algorithm can be verified.

The managerial insights of this study can be summarized as follows. Traditional routing problems restrict that a customer can only be served exactly once. This restriction is reasonable when the following two conditions hold. 1) The interval of the time window is tight. 2) The vehicle capacity is relatively large compared to the customer demands. In such cases, the split option would have very little effect according to our findings. However, in several practical applications such as water sparkling service, the split delivery must be considered, and it could significantly lower the total traveling cost. In practice, the routing optimization problem is subject to various kinds of constraints and uncertainty issues. Although our approach could be considered as an excellent solver for SDCARPTW, there are still gaps between algorithmic development and systematic deployment.

To extend our work, there should be more powerful algorithms that can be developed. For example, we can apply ejection pool, guided local search or other techniques, which have demonstrated their capabilities in handling time window constraints. We can also develop more sophisticate search operators to better handle the split-delivery option.

REFERENCES

[1] F. Wang, "Parallel control and management for intelligent transportation systems: Concepts, architectures, and applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 630–638, 2010.

[2] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "Finding the shortest path in stochastic vehicle routing: A cardinality minimization approach," *IEEE Transactions*

on Intelligent Transportation Systems, vol. 17, no. 6, pp. 1688–1702, 2016.

[3] X. Wang, T. Choi, H. Liu, and X. Yue, "Novel ant colony optimization methods for simplifying solution construction in vehicle routing problems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3132–3141, 2016.

[4] A. Hertz, G. Laporte, and M. Mittaz, "A tabu search heuristic for the capacitated arc routing problem," *Operations Research*, vol. 48, no. 1, pp. 129–135, 2000.

[5] P. A. Mullaseril, M. Dror, and J. Leung, "Split-delivery routeing heuristics in livestock feed distribution," *Journal of the Operational Research Society*, vol. 48, no. 2, pp. 107–116, 1997.

[6] B. L. Golden and R. T. Wong, "Capacitated arc routing problems," *Networks*, vol. 11, no. 3, pp. 305–315, 1981.

[7] H. A. Eiselt, M. Gendreau, and G. Laporte, "Arc routing problems, part i: The chinese postman problem," *Operations Research*, vol. 43, no. 2, pp. 231–242, 1995.

[8] ——, "Arc routing problems, part ii: The rural postman problem," *Operations Research*, vol. 43, no. 3, pp. 399–414, 1995.

[9] L. Santos, J. Coutinho-Rodrigues, and J. R. Current, "An improved ant colony optimization based algorithm for the capacitated arc routing problem," *Transportation Research Part B: Methodological*, vol. 44, no. 2, pp. 246 – 266, 2010.

[10] M. Liu, H. K. Singh, and T. Ray, "Application specific instance generator and a memetic algorithm for capacitated arc routing problems," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 249 – 266, 2014, special Issue on.

[11] G. Laporte, R. Musmanno, and F. Vocaturo, "An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands," *Transportation Science*, vol. 44, no. 1, pp. 125–135, 2010.

[12] E. Bartolini, J.-F. Cordeau, and G. Laporte, "Improved lower bounds and exact algorithm for the capacitated arc routing problem," *Mathematical Programming*, vol. 137, no. 1-2, pp. 409–452, 2013.

[13] C. Bode and S. Irnich, "Cut-first branch-and-price-second for the capacitated arc-routing problem," *Operations Research*, vol. 60, no. 5, pp. 1167–1182, 2012.

[14] R. Baldacci and V. Maniezzo, "Exact methods based on node-routing formulations for undirected arc-routing problems," *Networks*, vol. 47, no. 1, pp. 52–60, 2006.

[15] H. Longo, M. P. De Aragão, and E. Uchoa, "Solving capacitated arc routing problems using a transformation to the cvrp," *Computers & Operations Research*, vol. 33, no. 6, pp. 1823–1837, 2006.

[16] W. Ramdane-Cherif, "Evolutionary algorithms for capacitated arc routing problems with time windows," *IFAC Proceedings Volumes*, vol. 39, no. 3, pp. 321–326, 2006.

[17] M. Reghioui, C. Prins, and N. Labadi, "Grasp with path relinking for the capacitated arc routing problem with time windows," in *Workshops on Applications of Evolutionary Computation*. Springer, 2007, pp. 722–731.

[18] E. L. Johnson and S. Wøhlk, "Solving the capacitated arc routing problem with time windows using column generation," in *CORAL Working Paper L-2008-09*. University of Aarhus, 2009.

[19] C. Ciancio, D. Laganá, and F. Vocaturo, "Branch-price-and-cut for the mixed capacitated general routing problem with time windows," *European Journal of Operational Research*, vol. 267, no. 1, pp. 187–199, 2018.

[20] C. Archetti, M. G. Speranza, and A. Hertz, "A tabu search algorithm for the split delivery vehicle routing problem," *Transportation Science*, vol. 40, no. 1, pp. 64–73, 2006.

[21] S. Chen, B. Golden, and E. Wasil, "The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results," *Networks*, vol. 49, no. 4, pp. 318–329, 2007.

[22] C. Archetti, N. Bianchessi, and M. G. Speranza, "A column generation approach for the split delivery vehicle routing problem," *Networks*, vol. 58, no. 4, pp. 241–254, 2011.

[23] C. Archetti, M. Bouchard, and G. Desaulniers, "Enhanced branch and price and cut for vehicle routing with split deliveries and time windows," *Transportation Science*, vol. 45, no. 3, pp. 285–298, 2011.

[24] G. Desaulniers, "Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows," *Operations Research*, vol. 58, no. 1, pp. 179–192, 2010.

[25] N. Bianchessi and S. Irnich, "Branch-and-cut for the split delivery vehicle routing problem with time windows," *Transportation Science*, vol. 53, no. 2, pp. 442–462, 2019.

[26] Z. Luo, H. Qin, W. Zhu, and A. Lim, "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, pp. 668–687, 2017.

[27] N. Bianchessi, M. Drexl, and S. Irnich, "The split delivery vehicle routing problem with time windows and customer inconvenience constraints," *Transportation Science*, vol. 53, no. 4, pp. 1067–1084, 2019.

[28] J. Li, H. Qin, R. Baldacci, and W. Zhu, "Branch-and-price-and-cut for the synchronized vehicle routing problem with split delivery, proportional service time and multiple time windows," *Transportation Research Part E: Logistics and Transportation Review*, vol. 140, p. 101955, 2020.

[29] M. Dror and P. Trudeau, "Savings by split delivery routing," *Transportation Science*, vol. 23, no. 2, pp. 141–145, 1989.

[30] J.-M. Belenguer, E. Benavent, N. Labadi, C. Prins, and M. Reghioui, "Split-delivery capacitated arc-routing problem: Lower bound and metaheuristic," *Transportation Science*, vol. 44, no. 2, pp. 206–220, 2010.

[31] C. Archetti, M. G. Speranza, and M. W. Savelsbergh, "An optimization-based heuristic for the split delivery vehicle routing problem," *Transportation Science*, vol. 42, no. 1, pp. 22–31, 2008.

[32] Z. Zhang, H. He, Z. Luo, H. Qin, and S. Guo, "An efficient forest-based tabu search algorithm for the split-

delivery vehicle routing problem," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[33] M. M. Silva, A. Subramanian, and L. S. Ochi, "An iterated local search heuristic for the split delivery vehicle routing problem," *Computers & Operations Research*, vol. 53, pp. 234–249, 2015.

[34] S. Wøhlk, "Contributions to arc routing," in *Ph.D. thesis*. Faculty of Social Sciences, University of Southern Denmark, 2005.

Qidong Lai received B.S. degree in School of Data Science and Computer Science from Sun Yat-sen University, China in 2019. He is currently pursuing a Ph.D degree in Department of Management Sciences, City University of Hong Kong. His interests are arc routing problem and reinforcement learning.

Zizhen Zhang received B.S. and M.S. degree in the Department of Computer Science from Sun Yat-sen University, China, in 2007 and 2009, respectively. He received a Ph.D degree from City University of Hong Kong (2014). He is currently an Associate Professor in School of Data and Computer Science, Sun Yat-sen University, China. His research interests include computational intelligence and its applications in production, transportation and logistics.

Mingzhu Yu is currently an Associate Professor in the Institute of Big Data Intelligent Management and Decision of the College of Management in Shenzhen University. She received her Ph.D. from the Department of Industrial Engineering and Logistics Management in Hong Kong University of Science and Technology in 2012. Her research interests include transportation management and container terminal operations management. Her works have been published in IIE Transactions, Naval Research Logistics, European Journal of Operational Research, Transportation Research Part B, Transportation Research Part E, etc. She has engaged in several research projects sponsored by Chinese government.

Jiahai Wang (M'07-SM'19) received the Ph.D. degree in computer science from University of Toyama, Toyama, Japan, in 2005. In 2005, he joined Sun Yat-sen University, Guangzhou, China, where he is currently a Professor with the Department of Computer Science. His main research interests include computational intelligence and its applications.