Two-echelon urban deliveries using autonomous vehicles

Shaohua Yu, Jakob Puchinger, Shudong Sun

Two-echelon urban deliveries using autonomous vehicles

Shaohua Yu*[a,b], Jakob Puchinger[b,c], Shudong Sun[a]

[a] *Department of Industrial Engineering, Northwestern Polytechnical University, Xi'an 710072, China*
[b] *Laboratoire Genie Industriel, CentraleSupelec, Universite Paris-Saclay, Gif-sur-Yvette, France*
[c] *Institut de Recherche Technologique SystemX, Palaiseau, France*

Abstract

We introduce a two-echelon urban delivery problem with second-level unmanned vehicles. This problem typically applies for delivering parcels or other small commodities to pedestrianized areas such as campuses or residential clusters. To model the proposed vehicle routing problem, we introduce a mixed-integer program. We further propose construction heuristics and a hybrid metaheuristic approach with backtracking for solving larger instances. A sensitivity analysis for vehicle speed combinations reveals that increasing small autonomous vehicle (SAV) speeds has only very limited effects on cost. We therefore recommend to keep SAV speeds rather low because of a more pedestrian friendly environment in practical implementations.

Keywords: Two-echelon vehicle routing, Urban deliveries, Autonomous vehicles, Hybrid metaheuristics

1. Introduction

1.1. Urban deliveries

Over 50% of the world's population is now living in cities. In Europe, around 75% of the population already lives in urban areas (European Commission 2014). Optimizing urban transport has become an important part of transport management as it will affect the quality of life of increasingly more urban citizens. Urban mobility accounts for 40% of all CO_2 emissions from road transport and up to 70% of other pollutants from transport (European Commission 2015). Cities are forced to regulate access to their most populous central districts by limiting access to various means of transport in an effort to address these transport and environmental issues.

Traditional urban deliveries are a critical component of urban transport, but they are criticized for causing traffic jams and urban pollution today. Hence, small electric vehicles are encouraged for distribution. In parallel, rising labor costs and restrictions on delivery-staff workhours are challenging delivery companies' efforts to provide customers with cheap, efficient and round-the-clock courier services, while the rapid development of e-commerce is driving a steady yet steep rise in parcel delivery demands. Cities thus need to identify new delivery strategies to increase the quality of life of their citizens while keeping traffic smooth and environmental pollution under control. Delivery companies need to reduce costs, improve customer satisfaction, and comply with labor rights. In this context, unmanned electric transportation represents an excellent choice for city logistics.

1.2. The application of new technologies in urban deliveries

By leveraging new technologies (Autonomous Vehicles, Drones, etc.), many companies are developing new logistic systems that can change the competition landscape (Tang and Veelenturf 2019). We observed interest in freight distribution with robots has surged in the past five years, producing a number of studies and tests on new automated logistics and distribution tools. For example, UPS tested home delivery via truck and drone in Florida as a move towards a more automated delivery process (Lithia 2017). JD.com launched city delivery by autonomous vehicles in

Email addresses: shaohua.yu@centralesupelec.fr (Shaohua Yu*), jakob.puchinger@irt-systemx.fr (Jakob Puchinger), sdsun@nwpu.edu.cn (Shudong Sun)

several Chinese universities and districts in Beijing (LI 2017, Gu 2018). French start-up TwinswHeel is testing an unmanned delivery robot with the ability to climb a certain sidewalk height to complete last-mile delivery (Chevallier 2017). Note that most small fully automated ground vehicles (SAV) are electrically driven, which will help reduce local emissions in cities.

Starship Technologies and JD.COM believe small autonomous vehicles are ideal for urban distribution and have implemented autonomous distribution on campuses and residential areas (Andrew 2019, Gu 2018). It seems that autonomous delivery services can be realized in certain areas as of today. Researchers have begun to study autonomous vehicle delivery problems under restricted conditions, for example Scherr et al. (2019) designed an urban service network with mixed autonomous fleets.

However, in the foreseeable future, the applications of the SAV are limited by its travel distances and speed for technical and safety reasons. Hence, one innovative city logistics delivery concept was presented by combining traditional vehicles and SAVs. Boysen et al. (2018) considered using large vehicles (LVs) to transport and drop off small autonomous vehicles. They presented a decentralized robot depot within the city center, only used for storing delivery robots. Mercedes-Benz also presented the idea and works closely together with Starship Technologies to develop the "mothership approach" (truck carries SAVs) in their Future Transportation unit. They believe that the carrier system can avoid the drawback of robot-only delivery (Daimler 2017).

Based on the mothership concept, we propose an LV-SAV model where multiple LVs cooperate with their associated SAVs. The LV carries the SAVs, sending out and picking up the SAV in rendezvous nodes while the SAV manages customer service. The LV-SAV model has the following advantages:

(1) The LV-SAV model can avoid some drawbacks of robot-only delivery. For example, the SAV will only have a very limited range. Logically, the range is limited by battery capacity, besides, the robots move at walking speed for safety reasons, so that their application for long distances is not efficient (Daimler 2017). If we adopt the LV-SAV model, the LV can carry SAVs to implement long distances and high-speed transportation. (2) The LV-SAV model can greatly reduce the cost of the enterprise in operational aspects when compared to the traditional two echelon delivery model. Since the LV-SAV model does not need suitable real city estate, also without a lot of manpower cost. Note that satellites and labor costs are two big expenses of logistics enterprises. (3) The number of parking nodes is always larger than that of satellites, which means LVs can be more flexible in choosing transfer nodes. Moreover, the second level route in the LV-SAV model is an open route, the SAV can choose being picked up from a different node than the one they have been dropped off. Which often brings about a lower cost at the second level of delivery. (4) Compared to the truck-based drone model, we believe the LV-SAV model is safer in urban delivery. For example, when a fast-flying drone stalls, it will cause serious accidents in a city. However, to a slowly moving SAV, accidents caused by stalled vehicles are controllable.

The contributions of this paper are as follows. We consider a new two-echelon urban delivery concept with time windows relying on autonomous vehicles, in which the *2nd-level route* is an open route. First we introduce the problem and propose a mathematical formulation. A column generation procedure is presented for trying to get a better lower bound for the MIP model. Next we propose a construction heuristic for the newly introduced problem, which is useful for quickly generating feasible initial solutions and providing a first upper bound. We then propose a multi-start hybrid metaheuristic approach based iterative local search and backtracking. The backtracking procedure is led to accurately connect the chosen SAV routes to the LV route. Furthermore, we analyze how LV/SAV speed combinations influence the objective value, which can provide a reference for real-world implementation of such a service.

The remainder of the paper is structured as follows. A brief literature review is provided in Section 2. Then, Section 3 describes the problem and model and Section 4 elaborates on the heuristic approaches. A computational study is presented in Section 5 and Section 6 concludes the paper.

2. Literature review

Vehicle routing problems (VRP) have been studied for nearly 60 years (Braekers et al. 2016). The aim of these kinds of models is to design the service network for a fleet of vehicles originating and ending at a depot to a set of customers via the shortest possible total travel distance. This section presents a literature review on variants of the VRP, especially those related to the proposed LV-SAV model. We also discuss the similarities and differences between our LV-SAV model and other related models.

The truck and trailer routing problem (TTRP), as shown in Figure 1, describes a fleet of truck and trailer combinations with known capacity serving a set of customers with pre-determined demands and locations. In this problem, a vehicle may be a truck pulling a trailer, called a complete vehicle, or a single truck, called a pure truck. Some customers must be served by a truck while others can be served either by a truck or a complete vehicle (Chao 2002, Li et al. 2016, Parragh and Cordeau 2017, Rothenbächer et al. 2018). In the LV-SAV model, both the LV and the SAV can move by themselves, whereas the trailer in TTPR model stays stationary while the truck is not pulling. Furthermore, in the LV-SAV model, only the SAV provides service to customers, whereas in the TTRP model both the pure truck and the complete vehicle can serve customers.

The two-echelon vehicle routing problem (2E-VRP), which is shown in Figure 2, is a well-known variant of the classic VRP. It involves determining a set of optimal routes for a two-level freight distribution system, where goods are delivered from a depot to a subset of intermediary satellites in the first echelon, and from the satellites to customers in the second echelon. For cost effectiveness reasons, delivery tasks in the first echelon are usually accomplished by large identical trucks while delivery tasks in the second echelon are usually accomplished by small identical vehicles (Baldacci et al. 2013, Dellaert et al. 2018, Liu et al. 2018). In 2E-VRP, the intermediary satellite can be seen as a transfer station. The freight is the primary connection between truck and small identical vehicle. In the LV-SAV model, the intermediary satellite is just like a rendezvous node and does not possess any goods storage function. Furthermore, the connection between LV and SAV is closer than that of truck and small vehicle in the 2E-VRP problem. The LV not only carries the SAVs but also sends them out and picks them up at same/different rendezvous nodes.

The two-echelon location routing problem (2E-LRP), which is shown in Figure 3, is similar to the 2E-VRP. The main difference is that the 2E-LRP incorporates the location decision problem into the VRP while the 2E-VRP does not (Crainic et al. 2011, Cuda et al. 2015, Bala et al. 2017, Wang et al. 2018). The LV-SAV model is like the 2E-LRP model in that we need to decide which LVs/SAVs need to visit which rendezvous nodes, but the models differ on the earliest time that a vehicle is allowed to leave the pick-up rendezvous node. In 2E-LRP, the earliest leaving time of a vehicle is equal to the time when the vehicle arrived at the satellites, whereas in the LV-SAV model it depends on the time when its SAVs arrive at the pick-up node.

The vehicle routing problem with pickup-and-delivery (VRPPD), depicted in Figure 4, is an essential family of routing problems in which freight or passengers have to be transported between different origins and destinations (Paolo and Vigo 2014). The VRPPD can basically be subdivided into four subclasses (Polat 2017). In the VRP with clustered backhauls (VRPCB), mixed linehauls and backhauls (VRPMB), and simultaneous pick-ups and deliveries (VRPSPD), the vehicles only visit each customer once for pickups and deliveries, i.e. loads cannot be split. In the VRP with divisible deliveries and pickups (VRPDDP), the vehicles can arrive at each customer twice. Nagy et al. (2013) formulated the VRPDDP as a mixed-integer linear programming problem and presented an exact and heuristic algorithm to implement the problem. Polat (2017) presented an efficient parallel approach based on variable neighborhood search to solve the VRPDDP that significantly improved the best solutions available in the literature.

Note that in the LV-SAV model, the LV picks up and deliveries SAVs, while the truck in VRPPD model picks up and deliveries freights. Besides, the LV must first launch the SAV before recovering it. Whereas the vehicle has no priority for delivery and pick-up in the VPRPD, they could be delivery first and pickup second; mixed pickups and deliveries; simultaneous pickups and deliveries. In addition, in the LV-SAV model there is a minimum time interval between releasing and recovering the same SAV. This interval equals the shortest completion time for the SAV to finish delivery to all its customers from origins to destinations.

The open vehicle routing problem (OVRP), in which a vehicle does not necessarily return to the start node after servicing the last customer on its route, performs as a Hamiltonian path (or Hamiltonian circuit if the vehicle chooses to come back to the start node) (Li et al. 2007, Repoussis et al. 2007, Brandão 2018). The OVRP has attracted less attention than the VRP, largely due to its limited application environment: the main application background cited in the academic study is a company using hired vehicles for goods delivery, where the vehicle does not need to return to the depot after finishing serving the last customer. Note that in the second-level LV-SAV delivery scenario, the SAV route could be regarded as a Hamiltonian path since the LV can drop off and pick up an SAV at different rendezvous nodes. Consequently, the OVRP can be expected to receive more attention as automated distribution such as automated vehicle and/or drone delivery gains currency. The OVRP is shown in Figure 5.

Hybrid truck and drone delivery, shown in Figure 6, is another configuration attracting increasing attention (Otto et al. 2018). In hybrid truck-drone delivery, the drone can take off from the roof of the truck to serve customers. After finishing home delivery, the drone flies back and lands on the roof of the truck. There are two types of hybrid

truck and drone delivery: one where the truck only serves as a mobile platform for launching and recovering the drone (Carlsson and Song 2017, Luo et al. 2017, Yu et al. 2018, Karak and Abdelghany 2019, Poikonen and Golden 2020), and a second system where the truck also serves customers (Murray and Chu 2015, Poikonen et al. 2017, Wang et al. 2017, Pugliese and Guerriero 2017, Bouman et al. 2018, Agatz et al. 2018, Schermer et al. 2019a,b, Karak and Abdelghany 2019, Wang and Sheu 2019, Sacramento et al. 2019, Poikonen et al. 2019, Murray and Raj 2019).

Our LV-SAV model is similar to the first system where the truck does not serve customers. However, in our model, we study multiple LVs carrying multiple SAVs with time windows, and each SAV can serve more than one customer. To the best of our knowledge, these characteristics have not been considered simultaneously in existing models. Furthermore, the SAV travels at a slower speed than the LV while the drone travels faster than the truck. Note that the SAV is a competitive option for city logistics and distribution whereas the drone is far more efficient for rural logistics. Table 1 provides similarities and differences between hybrid truck and drone problems that were mentioned. *Visit MC* represents the number of customers a drone can visit during one trip; *Customer TW* represents whether there is time window constraints for customers; *Truck delivery* represents whether the truck can serve customer directly.

Table 1: Similarities and differences between hybrid truck and drone problems

Reference	Trucks	Drones	Objective	Visit MC	Customer TW	Truck delivery	Contribution
Murray and Chu (2015)	1	1	time	1	no	yes	MILP, heuristic
Poikonen et al. (2017), Wang et al. (2017)	n	m	time	1	no	yes	Theoretical insights
Pugliese and Guerriero (2017)	n	m	cost	1	yes	yes	MILP
Luo et al. (2017)	n	1	time	m	no	no	MILP, heuristic
Carlsson and Song (2017)	1	1	time	1	no	no	Heuristic
Agatz et al. (2018)	1	1	time	1	no	yes	IP, heuristic
Bouman et al. (2018)	1	1	cost	1	no	yes	DP
Yu et al. (2018)	n	1	time	m	no	no	GLNS solver, ILP
Schermer et al. (2019a,b)	n	m	time	1	no	yes	MILP, VNS, matheuristic
Karak and Abdelghany (2019)	1	m	cost	m	no	no	MIP, heuristic
Sacramento et al. (2019)	n	1	cost	1	no	yes	MILP, ALNS
Wang and Sheu (2019)	n	m	cost	m	no	yes	MIP, branch-and-price
Poikonen et al. (2019)	1	1	time	1	no	yes	Branch-and-Bound
Murray and Raj (2019)	1	m	time	1	no	yes	MILP, heuristic
Poikonen and Golden (2020)	1	m	time	m	no	no	ILP, heuristic
This work	n	m	cost	m	yes	no	MILP, hybrid metaheuristic

Table 2 gives a roll-up of the similarities and differences between various two-echelon routing problems.

Table 2: Similarities and differences between various two-echelon routing problems

	TTRP	2E-VRP	2E-LRP	Truck-drone	LV-SAV
LV pick-up and delivery?	YES	NO	NO	YES	YES
Open SAV route?	NO	NO	NO	YES	YES
LV choose satellites?	NO	NO	YES	YES	YES
LV provide services?	YES	NO	NO	YES/NO	NO
LV speed >SAV speed?	/	/	/	NO	YES

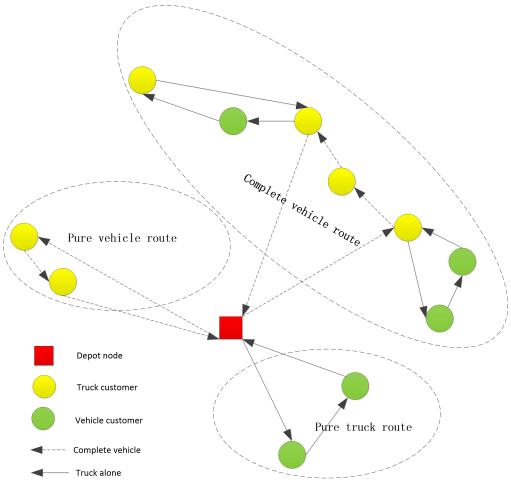

Figure 1: TTRP

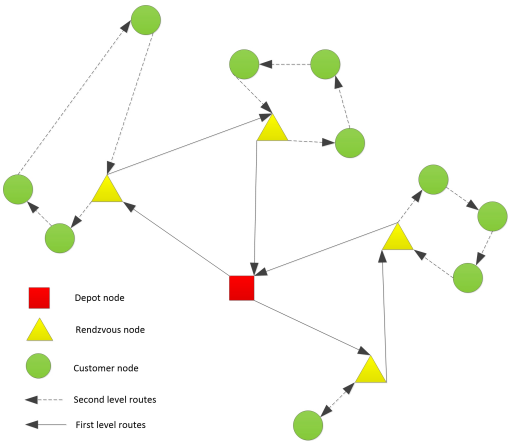

Figure 2: 2E-VRP

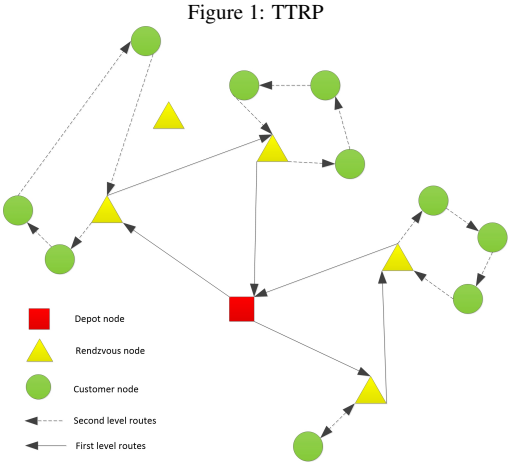

Figure 3: 2E-LRP

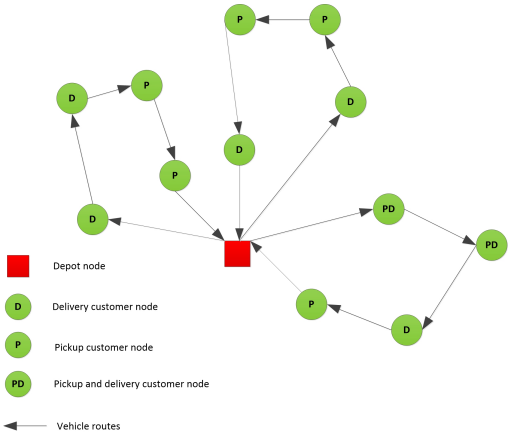

Figure 4: VRPPD

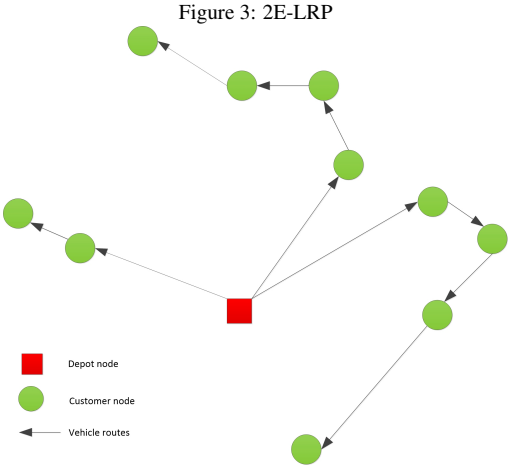

Figure 5: OVRP

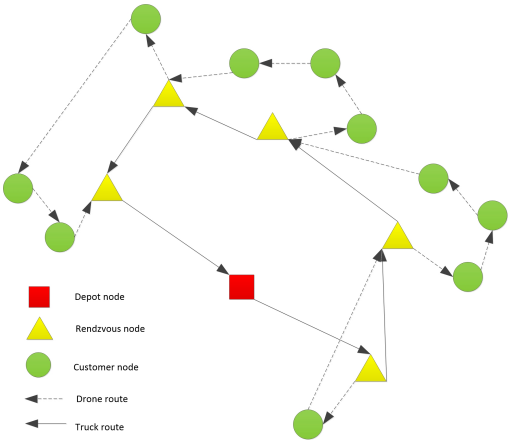

Figure 6: Truck-drone routing problem

3. Problem description and model

3.1. Problem statement

We consider a two-echelon urban delivery problem using SAVs for *2nd-level route* delivery. The LV carries the SAVs on the *1st-level route* and drops off and picks up them in the rendezvous nodes, while the SAV handles customer service on the *2nd-level route*. The research developed in this paper considers using the LV only for carrying SAVs, and so no direct shipping from LVs to customers is allowed. This setting is reasonable, since the SAV can only deliver parcels or other small commodities to pedestrianized areas such as campuses or residential clusters with current technologies. Also, our target customers are in these pedestrianized areas, where LVs are often banned. Hence, we assume the LV cannot serve customers directly.

Since campuses and residential clusters have multiple entrances and exits, the LV can drop off and pick up an SAV at different positions. In other words, the LV can move to other rendezvous nodes after making a drop-off operation without having to wait for its SAV to come back to the same rendezvous location. Besides, the SAV is not forced to return back to the rendezvous node it departed from, which means the *2nd-level route* is an open route.

Each pick-up or drop-off (rendezvous) node can only be visited at most once by the same vehicle. This setting is reasonable, as an LV can release all its associated SAVs immediately at a drop-off node and has no need to reach that node again. Likewise, an LV can arrive at a pick-up node when all its associated SAVs have arrived. Hence, a drop-off/pick-up node does not have to be visited twice by the same LV. Each customer node must be visited by just one SAV exactly once. In addition, customer nodes and depot have their time windows based on real demand in city logistics. Our model accommodates waiting at all locations without cost. Moreover, we allow an SAV to visit multiple customers during a dispatch rather than only visit one customer, since the maximum capacity for an SAV usually larger than that of a drone. The travel range of an LV is infinite. Nevertheless, the total travel time of an SAV cannot exceed a predetermined value on the *2nd-level route*, as the SAV tends to be small-sized and thus equipped with limited fuel/battery capacity.

We present three (simplifying) assumptions. In order to simplify the model and reduce the complexity of our problem, an SAV is dropped off and picked up by a single LV. Furthermore, we assume constant operation times, and we can therefore integrate it into the travel time from and to the rendezvous nodes (Grangier et al. 2016). In the LV-SAV model proposed here, freight can only be loaded when rounds begin at the depot. In other words, the LV carries a sufficient number of SAVs full of freight instead of carrying some SAVs and freight and then taking on freight replenishment during the LV route.

Figure 7 shows an example of the LV-SAV model. Triangles represent rendezvous nodes, the square represents the depot, and circles correspond to customer nodes. Solid lines correspond to *1st-level routes* (LV routes) and dotted lines correspond to *2nd-level routes* (SAV routes).

3.2. Mixed Linear Integer Programming Model

The problem is defined on a directed graph $G = (V, A)$, where the depot V_0 is represented by two nodes 0 and $0'$. Let $V_r = \{1, 2, ..., m, m+1, ..., 2m\}$ be rendezvous nodes where LVs drop off and pick up their SAVs. The node $m + i$ in pick-up nodes set $V_p = \{m + 1, m + 2, ..., 2m\}$ is a copy of the drop-off node i (physical rendezvous node) in set $V_d = \{1, 2, ..., m\}$. Note that node i and node $m + i$ correspond to a same physical rendezvous node. We use different names to distinguish drop-off and pick-up operations of LVs. Furthermore, $V_c = \{2m+1, 2m+2, ..., 2m+n\}$ represents the customer nodes set. Moreover, we define $V_{dc} = V_d \cup V_c$ and $V_{pc} = V_p \cup V_c$. Also, $V_r^0 = V_r \cup \{0\}$ and $V_c^0 = V_c \cup \{0\}$ while $V_r^{'} = V_r \cup \{0^{'}\}$ and $V_c^{'} = V_c \cup \{0^{'}\}$. Let $A_1 = \{(i, j) \mid i \in \{0\}; j \in V_d\} \cup \{(i, j) \mid i, j \in V_r, i \neq j\} \cup \{(i, j) \mid i \in V_p; j \in \{0^{'}\}\}$ be the *1st-level route* (LV routes) and $A_2 = \{(i, j) \mid i \in V_d; j \in V_c\} \cup \{(i, j) \mid i, j \in V_c, i \neq j\} \cup \{(i, j) \mid i \in V_c; j \in V_p\}$ be the *2nd-level route* (SAV routes).

For each edge, let $t_{i,j} > 0$ be the associated travel time and $c_{i,j}$ be the associated travel cost. The freight must be delivered from depot $\{0\}$ to customer node i with the demand d_i and serving time s_i. The time window of the customer nodes $i \in V_c$ is $[a_i, b_i]$, which is the time interval that the service at node i is allowed to start. Furthermore, let $[a_0, b_0] = [a_{0'}, b_{0'}]$, where $a_0/a_0^{'}$ represents the earliest possible departure time from the depot $0/0'$ and $b_0/b_0^{'}$ is the latest possible arrival time at the depot $0/0'$. These time windows are hard constraints. $F_T = \{1, 2, ..., K\}$ is the set of LVs and $F_D^k = \{(k, 1), (k, 2), ..., (k, L)\}$ is the set of SAVs belonging to the *kth* LV, where K is the number of LVs and L is a maximum number of SAVs assignable to an LV. However, this does not mean each LV has to carry L SAVs. $F_D = F_D^1 \cup F_D^2 \cup, ..., \cup F_D^K = \{(1, 1), (1, 2), ..., (k, l), ..., (K, L)\}$ denotes the set of all SAVs. Let C, T, and M be maximum

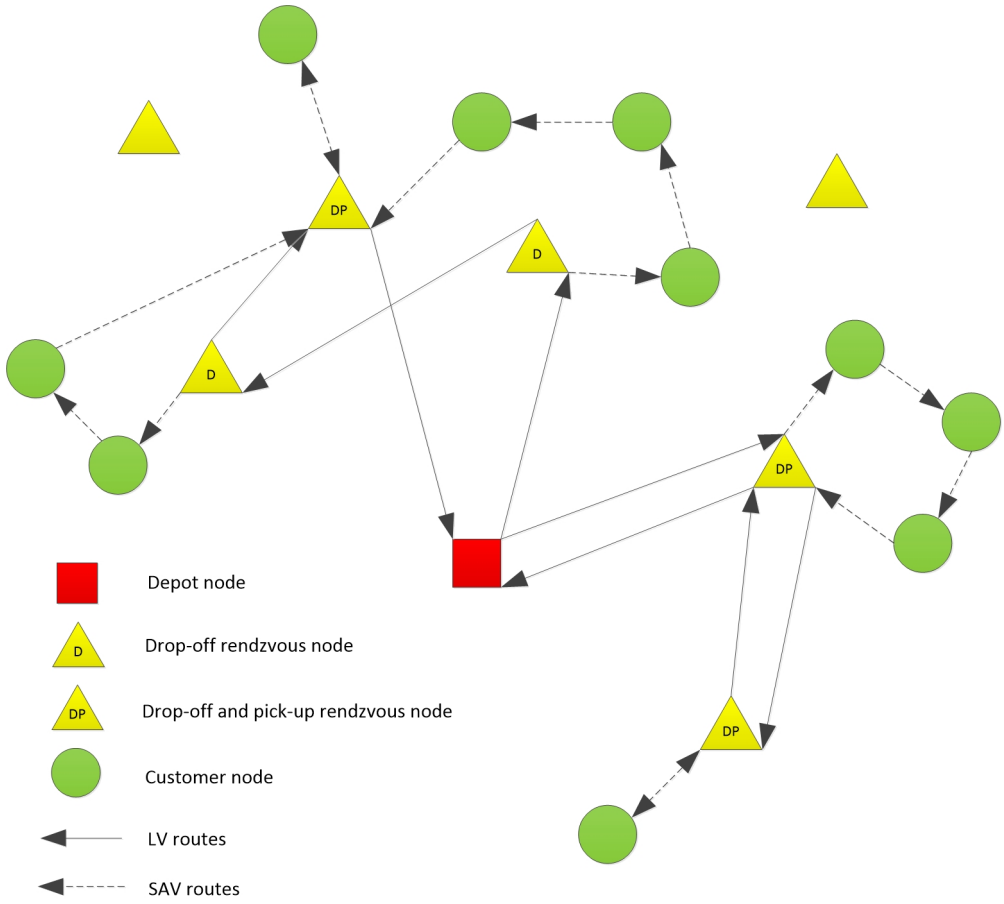

Figure 7: Delivery system with LV/SAV

capacity, maximum total travel time for an SAV, and an arbitrary large constant number, respectively. In addition, we further introduce the following decision variables:

- Let $x_{i,j,k}$ equal to 1 if arc (i, j) in A_1 is traveled by the *kth* LV, 0 otherwise.
- Let $y_{i,j}^{k,l}$ equal to 1 if arc (i, j) in A_2 is traveled by the *lth* SAV belonging to *kth* LV, 0 otherwise.
- Let $Q_{i,j,k}$ be the SAV flow carried by *kth* LV passing through arc (i, j) in A_1, i.e., the number of SAVs.
- Let W_i^k be the arrival time of *kth* LV (or the SAV belonging to the *kth* LV) at node i. Note that W_i^k represents the last arrival time of *kth* LV and the SAVs belonging to the *kth* LV at node i.

The LV-SAV problem is modeled as the following MIP:

$$\text{Lex-min}\Big(\sum_{k\in F_T}\sum_{j\in V_d} x_{0,j,k}, \sum_{k\in F_T}\sum_{(i,j)\in A_1} c_{i,j}x_{i,j,k} + \sum_{(k,l)\in F_D}\sum_{(i,j)\in A_2} c_{i,j}y_{i,j}^{k,l}\Big) \tag{1}$$

$$\sum_{(i,j)\in A_1} x_{i,j,k} \le 1, \qquad \forall j \in V_r^{'}, k \in F_T \tag{2}$$

$$\sum_{i\in V_p} x_{i,0',k} = \sum_{j\in V_d} x_{0,j,k}, \qquad \forall k \in F_T \tag{3}$$

$$\sum_{(i,j)\in A_1} x_{i,j,k} - \sum_{(j,i)\in A_1} x_{j,i,k} = 0, \qquad \forall j \in V_r, k \in F_T \tag{4}$$

$$\sum_{(k,l)\in F_D} \sum_{i\in V_{dc}} y_{i,j}^{k,l} = 1, \qquad \forall j \in V_c \tag{5}$$

$$\sum_{i\in V_{dc}} y_{i,j}^{k,l} - \sum_{i\in V_{pc}} y_{j,i}^{k,l} = 0, \qquad \forall j \in V_c, (k,l) \in F_D \tag{6}$$

$$\sum_{i\in V_p} Q_{i,0',k} = \sum_{j\in V_d} Q_{0,j,k}, \qquad \forall k \in F_T \tag{7}$$

$$\sum_{j\in V_d} Q_{0,j,k} = \sum_{l\in F_D^k} \sum_{i\in V_c} \sum_{j\in V_d} y_{j,i}^{k,l}, \qquad \forall k \in F_T \tag{8}$$

$$\sum_{i\in V_r^0} Q_{i,j,k} - \sum_{i\in V_r} Q_{j,i,k} = \sum_{l\in F_D^k} \sum_{i\in V_c} y_{j,i}^{k,l}, \qquad \forall j \in V_d, k \in F_T \tag{9}$$

$$\sum_{i\in V_r} Q_{i,j,k} - \sum_{i\in V_r^{'}} Q_{j,i,k} = -\sum_{l\in F_D^k} \sum_{i\in V_c} y_{i,j}^{k,l}, \qquad \forall j \in V_p, k \in F_T \tag{10}$$

$$0 \le Q_{i,j,k} \le x_{i,j,k} * L, \qquad \forall (i,j) \in A_1, k \in F_T \tag{11}$$

$$W_i^k + t_{i,j} - W_j^k \le M(1 - x_{i,j,k}), \qquad \forall (i,j) \in A_1, k \in F_T \tag{12}$$

$$W_i^k + t_{i,j} - W_j^k \le M(1 - y_{i,j}^{k,l}), \qquad \forall i \in V_d, j \in V_c, (k,l) \in F_D \tag{13}$$

$$W_i^k + t_{i,j} + s_i - W_j^k \le M(1 - y_{i,j}^{k,l}), \qquad \forall i \in V_c, j \in V_{pc}, (k,l) \in F_D \tag{14}$$

$$a_i \le W_i^k, \qquad \forall i \in V_c^0, k \in F_T \tag{15}$$

$$W_i^k \le b_i, \qquad \forall i \in V_c^{'}, k \in F_T \tag{16}$$

$$\sum_{i\in V_c} d_i \sum_{j\in V_{pc}} y_{i,j}^{k,l} \le C, \qquad \forall (k,l) \in F_D \tag{17}$$

$$\sum_{(i,j)\in A_2} y_{i,j}^{k,l} t_{i,j} \le T, \qquad \forall (k,l) \in F_D \tag{18}$$

$$x_{i,j,k} \in \{0,1\}, \qquad \forall (i,j) \in A_1, k \in F_T \tag{19}$$

$$y_{i,j}^{k,l} \in \{0,1\}, \qquad \forall (i,j) \in A_2, (k,l) \in F_D \tag{20}$$

The lexicographic objective function (1) minimizes the number of LVs first and then minimizes the total transportation cost of *1st-level* and *2nd-level* routes. Constraint (2) implies that each LV departs from (or arrives at) the rendezvous node no more than once, and each LV arrives at the depot no more than once. Constraints (3-4) ensures the number of arrivals is equal to the number of departures for the LV at the depot/rendezvous node. Constraints (5-6) indicate that each customer node is visited exactly once. Constraints (7-8) assure the number of SAVs (carried by the *kth* LV) departing from/arriving at the depot is equal to the total number of SAVs (carried by the *kth* LV) departing from the drop-off nodes. Constraints (9-10) link the number of SAVs carried by LV to the number of SAVs departing from the drop-off node/arriving at the pick-up node. Constraint (11) guarantees that the number of SAVs carried by a given LV cannot exceed its maximum capacity on *1st-level route*.

Constraint (12) is the time-flow constraint for the LVs. Constraints (13-14) are time-flow constraints for the SAVs. Service time does not need to be considered when an SAV departs from drop-off nodes to customer nodes, but need to be considered when an SAV is traveling between customer nodes or from customer nodes to pick-up nodes. Constraint

(12-14) can also eliminate the subtour of *1st-level route* and *2nd-level route*. Constraints (15-16) are the time window constraints for the customer nodes. Constraint (17) ensures the demand of every customer is met. Constraint (18) forces the maximum travel time of each SAV. Constraints (19-20) are the constraints on variables.

4. Methodology

This section introduces approximate solution methods for the LV-SAV problem. Section 4.1 proposes a construction heuristic to obtain a feasible solution quickly. The construction heuristic is applied to provide an upper bound during optimization of the primary objective of the MIP model, and also to generate multiple initial solutions for a hybrid metaheuristic approach. In Section 4.2 we propose a hybrid multi-start metaheuristic including destroy and repair operators together with a backtracking component.

4.1. Construction heuristic

The general structure of the construction heuristic is sketched out in Algorithm 1. Let S_2 be the set of rendezvous nodes and the depot. The sequence *GetStartCustomerNode*, *GetDropOffNode* and *ModifiedNearestNeighbor* is repeated to construct multiple SAV routes (*2nd-level route*) until no customer nodes are left in the unvisited customer nodes set S_1 (lines 2-8). Meanwhile, set S_1 and the complete SAV route set S_3 are updated (lines 6-7). Afterwards, a simple connection heuristic (*SimpleConnectionHeuristic*) is applied to construct multiple LV routes (*1st-level route*) to get the final solution S (line 9).

Algorithm 1 $Construction\_Heuristic(S_1, S_2)$

1: **Initialization:** $S_3 \leftarrow \varnothing$
2: **while** $S_1 \neq \varnothing$ **do**
3: $Node_1 \leftarrow GetStartCustomerNode(S_1)$
4: $Node_2 \leftarrow GetDropOffNode(Node_1, S_2)$
5: $SAVRoute \leftarrow ModifiedNearestNeighbor(Node_1, Node_2, S_1, S_2)$
6: Remove Customer nodes in $SAVRoute$ from S_1
7: $S_3.add(SAVRoute)$
8: **end while**
9: $S \leftarrow SimpleConnectionHeuristic(S_3)$
10: **Output:** S

Note: S_1: unvisited customer nodes, S_2: rendezvous nodes and depot, S_3: complete SAV routes, S: final solution.

First, we introduce how to construct the *2nd-level route*. The *GetStartCustomerNode* procedure is run to select a customer node for the second level route construction algorithm (lines 2-8) to start in Algorithm 1. Here, we introduce the deterministic/random start customer node selection strategies that can be used in *GetStartCustomerNode*. The deterministic start customer node selection strategy is the one we obtain the start customer node with the minimum latest service time (similar to (Li et al. 2016)), while in the random start customer node selection strategy, the start visiting customer node from the customer nodes set is chosen randomly to maintain the diversity of solutions.

After choosing a start customer node, we select a drop-off node to connect the depot, drop-off node and start customer node. We propose optimal/nearest drop-off node selection strategies that can be implemented in *GetDropOffNode*. The optimal drop-off node selection strategy chooses the drop-off node for which distance to depot plus distance to start customer node is the smallest, whereas the nearest drop-off node selection strategy picks the drop-off node with the nearest distance to the start customer node.

It is important to note here that the optimal drop-off node selection strategy tends to select a drop-off node close to the depot; whereas the nearest drop-off node selection strategy tends to choose a drop-off node close to customer nodes. These two different selection strategies thus yield a significant difference in initial solutions.

Next, we apply *ModifiedNearestNeighbor* to select the remaining customer nodes and a pick-up node sequentially for constructing the SAV route. *ModifiedNearestNeighbor* not only uses the nearest distance information like the nearest neighbor search does, but it also considers the start time window information at the chosen customer node.

The total nearest of *ModifiedNearestNeighbor* is described as the SAV start serving time earliest. In other words, we choose the customer node with the earliest time at which the SAV can start to provide services.

The specific operation of *ModifiedNearestNeighbor* chooses the total nearest customer node based on the current chosen point subject to all constraints. The constraints involve the time window constraints for the SAV and LV, the capacity constraint, and the maximum travel time constraint for the SAV. The procedure iterates until there is no customer node that satisfies the constraints, then selects the nearest pick-up node to finally make a complete SAV route.

Second, we introduce how the *SimpleConnectionHeuristic* constructs the *1st-level route*. The general structure of the simple connection heuristic is sketched out in Algorithm 2, which is based on a distance nearest neighbor search to connect the given SAV routes with LVs. The sequence of *GetStartRendezvousNode*, *GetSAVRoute* and *ConnectionCheck* procedure is repeated to construct one complete LV-SAV route until the number of chosen SAV routes in set S_2 is equal to the capacity of the LV (LV_{max}) or until all unconnected SAV routes in set S_3 (a copy of unconnected SAV routes set S_1) have been tried (lines 5-15). The process iterates until all SAV routes are connected (lines 2-18). Finally, we output final solution S.

Algorithm 2 $Simple\_Connection\_Heuristic(S_1)$

1: **Initialization:** $S_2 \leftarrow \varnothing$, $S \leftarrow \varnothing$, LV_{max}
2: **while** $S_1 \neq \varnothing$ **do**
3: $S_2 \leftarrow \varnothing$
4: $S_3 \leftarrow Copy(S_1)$
5: **while** $S_3 \neq \varnothing$ **do**
6: $Node \leftarrow GetStartRendezvousNode(S_2, S_3)$
7: $SAVRoute \leftarrow GetSAVRoute(Node, S_2, S_3)$
8: $S_3.remove(SAVRoute)$
9: **if** $!(LVSAVRoute \leftarrow ConnectionCheck(S_2, SAVRoute))$ **then**
10: $S_2.add(SAVRoute)$
11: **end if**
12: **if** $len(S_2) = LV_{max}$ or $S_3 = \varnothing$ **then**
13: $S.add(LVSAVRoute)$, break while loop
14: **end if**
15: **end while**
16: $S_1.remove(S_2)$
17: **end while**
18: **Output:** S

Note: S_1: unconnected SAV routes set, S_2: chosen SAV routes, S_3: a copy of S_1, S: final solution, LV_{max}: capacity of LV.

The *GetStartRendezvousNode* procedure is applied to choose a drop-off node for Algorithm 2 to start constructing one LV-SAV route. If it is the first drop-off node we will choose, then we select the node with the maximum number of SAV routes departing from it. Otherwise, we select the drop-off node based on the nearest distance criterion. Furthermore, a drop-off node can be selected more than once if there are SAV routes from this node not chosen. The *GetSAVRoute* procedure is implemented to choose an SAV route based on the selected drop-off node. We classify the procedure into two types involving four sub-scenarios as described in Table 3. The *ConnectionCheck* procedure is executed to connect the chosen SAV routes. In order to simplify the connection process, *ConnectionCheck* accesses all the drop-off nodes first according to the chosen sequence. Afterward, *ConnectionCheck* visits the pick-up nodes according to the selected order. The procedure checks the constraints during the whole process.

Third, we give a simple example of how the simple connection heuristic performs in Figure 8. The triangles represent the rendezvous nodes, the square represents the depot, and the circles are the customer nodes. The solid line with the arrow is the LV route, and the dotted lines with the arrow are SAV routes. In order to simplify the example and focus on the specific selection process, we assume that the LV route 0-A-D-B-C-0 is feasible in advance. Also, we assume SAV route 1 has the shortest SAV arrival time at its pick-up node.

Table 3: Get SAV Route procedure

Scenario	Sub-Scenario	Operator
The drop-off node is the first node an LV visited in one LV-SAV route	There is no previous SAV route connected	Choose the corresponding SAV route with the smallest SAV arrival time at its pick-up node
	There is previous SAV route connected	Choose the node with nearest distance from the chosen SAV route's pick-up node to the previous SAV route's pick-up node
The drop-off node is not the first node an LV visited in one LV-SAV route	For the SAV routes departing from current drop-off node, there exist corresponding pick-up nodes the same as the pick-up nodes of already chosen SAV routes	Select the SAV route with the same pick-up node to the already chosen SAV routes
	For the SAV routes departing from current drop-off node, there not exist corresponding pick-up nodes the same as the pick-up nodes of already chosen SAV routes	Select the SAV route with the nearest distance between its pick-up node and the previous SAV route's pick-up node

- Step 1: First we choose drop-off node A as it has 3 SAVs departing from it, which is larger than the number for drop-off node D. Then we select SAV route 1 since it has the smallest SAV arrival time at its pick-up node. Afterward, we check LV route 0-A-B-0 and find it feasible.
- Step 2: We select drop-off node A since it is closest to the drop-off node of the previously chosen route 1. Next, we choose SAV route 2 since its pick-up node B is nearest to the pick-up node for the previously selected route 1. Finally, LV route 0-A-B-0 is connected, and it is feasible.
- Step 3: We select the drop-off node A and SAV route 3 based on the nearest distance criterion, as done in step 2. Afterward, we use *ConnectionCheck* to check and connect LV route 0-A-B-C-0.
- Step 4: First we find drop-off node D as it has the nearest distance to the drop-off node of the chosen route 3. Afterward, we choose SAV route 4 since its pick-up node C has already been chosen before. LV route 0-A-D-B-C-0 is feasible, and we find the number of SAVs that the LV carried is up to its maximum capacity. One complete LV-SAV route is successfully constructed.

4.2. Hybrid metaheuristic

Hemmelmayr et al. (2012) and Breunig et al. (2016) proposed a large neighborhood search-based approach for handling general two-echelon routing problems in which they used a destroy and repair operator with local search phase. Here, we draw on some of their ideas and then present a hybrid metaheuristic approach to improve the construction heuristic solutions. One way to achieve diversification is to re-start the procedure from a new solution (Martí et al. 2019). It is clear that if the iteration number is fixed, the initial solution has a significant impact on quality of the final solution. Hence, we also borrow some of the ideas used in multi-start heuristics (Nguyen et al. 2012, Han and Chu 2016) to design the hybrid metaheuristic.

The general structure of the hybrid metaheuristic is depicted in Figure 9. The destroy-repair loop aims to minimize the primary objective while the iterated local search (ILS) loop optimizes the primary and secondary objectives simultaneously. A multi-start loop restarts the algorithm by starting from a new initial solution clearly distinct from the previous one. Furthermore, we use maximum iteration numbers as the acceptance criteria 1,2,3,4.

A more detailed algorithm in pseudocode is given in Algorithm 3. The construction heuristics are used to generate multiple initial solutions from $S_1^{initial}$ to $S_{max}^{initial}$ (line 2). The destroy-repair loop with an iteration number $IterN2_{max}$

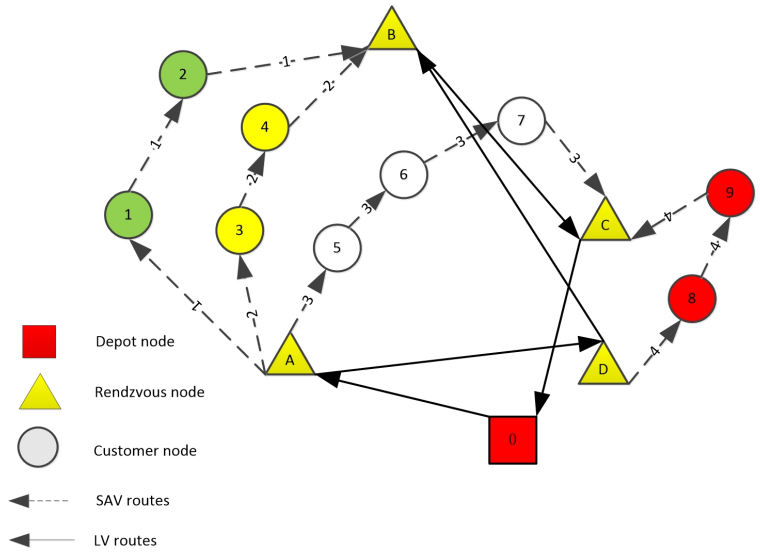

Figure 8: Simple Connection Heuristic example

(lines 5-13) is performed to obtain solution S^{dr} (line 14). In which the destroy and repair procedures are run until there is no customer node in the chosen customer node set S_{node} (lines 8-11). If we get S^{dr} for the first time or if the number of complete LV-SAV routes in S^{dr} is smaller than the previous one, the local search and ILS procedures are implemented (lines 16-27). Otherwise, the algorithm breaks out of the repair-destroy-ILS loop (lines 3- 32). The ILS loop (lines 17-26) is run to update and obtain best solution S^{best} (line 22). In which the perturbation and local search repeat until the iteration number is equal to $IterN3_{max}$. If the perturbation is feasible for the current best solution S^{best}, the local search is conducted to get solution S^{end} (lines 19-20). If S^{end} is better than S^{best}, then update S^{best} (lines 21-22). These procedures restart using the multi-start loop (lines 2-34), and every best solution S^{best} will be saved in solution set S (line 33).

For two-echelon routing problems, the number of satellites is generally much less than the number of customer nodes since a satellite is essentially like a small depot and maintaining its normal operation requires funds. Hence, using an exact algorithm to build the *1st-level route* is a viable method if the number of rendezvous nodes is small. In this paper, we first construct the *2nd-level route* and then use LVs to connect them. If we have several SAVs belonging to one LV, we can employ a backtracking algorithm to connect the SAV routes with the LV route. The method is applied in the repair and ILS procedures. After a customer node has been inserted into a complete LV-SAV route or after performing moves in the ILS, the backtracking algorithm is applied to check whether the *1st-level route* can be successfully connected, and then the backtracking outputs the solution with the minimum travel cost of *1st-level route* if needed.

4.2.1. Destroy and Repair

A feasible and straightforward way to reduce the number of LVs used is to choose a complete LV-SAV route to destroy it and then insert its customer nodes into other LV-SAV routes.

The destroy procedure destroys a chosen complete LV-SAV route into the list of nodes for re-inserting. In this paper, we consider randomly choosing an LV-SAV route to be destroyed, which guarantees a diversity of solutions. At each repair phase, we randomly insert the lists of nodes obtained by the destroy procedure to other complete LV-SAV routes in random order. In addition, the repair procedure inserts each customer node at its first feasible position or at the position with the largest positive saving of the secondary objective. If there are still nodes that cannot be successfully inserted at the end, the insertion fails.

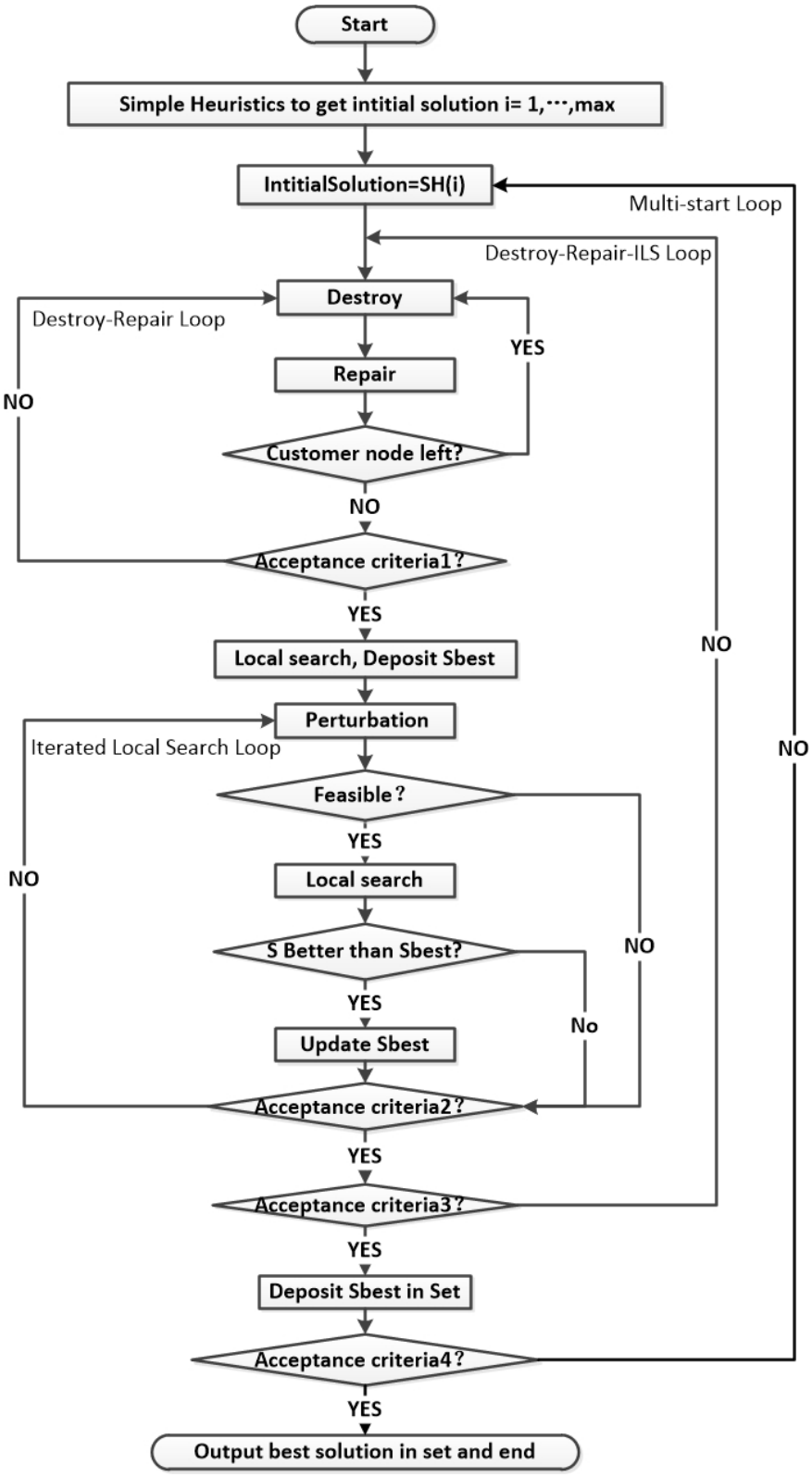

Figure 9: Hybrid metaheuristic-General flowchart

Algorithm 3 *Hybrid_metaheuristic*($IterN1_{max}$, $IterN2_{max}$, $IterN3_{max}$)

$S_1^{initial}, ..., S_{max}^{initial} \leftarrow ConsrtuctionHeuristic()$, The solution set $S \leftarrow \varnothing$
for $S_1^{initial}$ to $S_{max}^{initial}$ **do**
  Initialization: $IterN1 \leftarrow 0$
  while $IterN1 < IterN1_{max}$ **do**
    Initialization: The temp solution set $S^{temp} \leftarrow \varnothing$, $IterN2 \leftarrow 0$
    while $IterN2 < IterN2_{max}$ **do**
      $S_{node} \leftarrow \varnothing$, $S^{temp1} \leftarrow S_i^{initial}$
      while S_{node} is empty **do**
        $S^{temp2} \leftarrow S^{temp1}$
        $(S_{node}, S^{temp1}) \leftarrow Repair(Destroy(S^{temp1}))$
      end while
      $S^{temp}.add(S^{temp2})$, $IterN2 \leftarrow IterN2 + 1$
    end while
    Get S^{dr} by choosing the solution with minimum primary objective value in set S^{temp}
    if we get the S^{dr} for the first time or the number of complete LV-SAV route in S^{dr} is smaller than the previous one **then**
      $S^{best} \leftarrow Localsearch(S^{dr})$
      Initialization: $IterN3 \leftarrow 0$
      while $IterN3 < IterN3_{max}$ **do**
        if $Perturbation(S^{best})$ is feasible **then**
          $S^{end} \leftarrow Localsearch(Perturbation(S^{best}))$
          if S^{end} better than S^{best} **then**
            $S^{best} \leftarrow S^{end}$
          end if
        end if
        $IterN3 \leftarrow IterN3 + 1$
      end while
      $S_i^{initial} \leftarrow S^{best}$
    else
      break while loop
    end if
    $IterN1 \leftarrow IterN1 + 1$
  end while
  $S.add(S^{best})$
end for
Output: Output the best solution in S

4.2.2. Local search

In this paper, we exploit well-known moves such as insertion, swap, and 2-opt. Furthermore, the moves we called 'change-satellites' are also used to change the drop-off and pick-up nodes of an SAV route.

The 2-opt moves are executed inside the SAV route. Details on 2-opt can be found in (Croes 1958). The insertion and swap operators are applied in three different route configurations that are inside an SAV route, inside a complete LV-SAV route, and between complete LV-SAV routes, respectively. The insertion moves searches the intercalation of one node after one of the neighbor nodes while the swap moves explore swapping one node with one of the neighbor nodes. For one SAV route, the change-satellites operator involves changing the drop-off and pick-up node individually or simultaneously. Specific insertion, swap, and change-satellites operations are introduced in greater depth below.

Figure 10 describes three neighborhood structures, and each contains several sub-neighborhood structures. The insertion operators N1, N2, and N3 all consist of removing a customer node from a position i and inserting it after a position j: N1 removes and inserts customer nodes in the same SAV route, N2 removes and inserts customer node in different SAV routes from a same LV, and N3 removes and inserts customer node in different SAV routes between different LV-SAV routes. Swap operators N4, N5, and N6 all involve swapping customer-node positions. N4 swaps two customer nodes in the same SAV route whereas N6 swaps two customer nodes in different SAV routes incident to different LV-SAV routes. N5 swaps customer nodes in two SAV routes within a same LV route. The change-

satellites operators N7, N8, and N9 change pick-up node only, drop-off node only, and both drop-off and pick-up nodes, respectively.

Since the insertion moves make it possible to reduce the number of LV routes, the local search procedure can simultaneously optimize both the primary and secondary objectives. In the insertion process, we multiply a significant penalty by the number of LV routes in the cost function. Hence, once a solution emerges that reduces the primary objective, it can readily become the current optimal solution and can be saved.

In this paper, we apply the sequential Variable Neighbor Descent (VND) with first improvement (Duarte et al. 2018) to conduct the local search. Note that the neighbor sequence is randomly predetermined before starting the VND in order to promote a diversity of solutions.

4.2.3. Perturbation

We present the perturbation procedure involving three perturbations that bring a significant change to the structure of the solution, which is good for escaping local optima.

The first perturbation *Change-Multisatellites* is an enhanced version of the change-satellites operator which allows simultaneous change of two satellites in an LV-SAV route. The second perturbation *Change-SAVroute* determines that SAV routes can swap between different LV-SAV routes. The last perturbation *Destroy-Repair-Reconstruction* randomly destroys an LV-SAV route and inserts its customer nodes into other LV-SAV routes. If there are nodes left, the route is reconstructed by construction heuristics.

For the reconstruction procedure, we use the *GetStartCustomerNode* procedure by choosing the random start customer node selection strategy. Moreover, we randomly choose a drop-off node selection strategy in the *GetDropOffNode* procedure, which is used in the multi-start procedure of the hybrid metaheuristic. These two node selection strategies guarantee the diversity of solutions.

4.2.4. Backtracking algorithm for connection

In general, one LV carries several SAVs not exceeding its capacity. Furthermore, each SAV route starts from a drop-off node and ends at a pick-up node. However, different SAV routes may start from the same drop-off node or end at the same pick-up node. If the number of SAVs carried by an LV is larger than the number of physical rendezvous nodes, in other words, $L > m$, there are $2m$ nodes that will be visited at most by one LV. Otherwise, there are at most $2L$ nodes that will be visited by one LV. Overall, a maximum of $2 * min(L, m)$ different rendezvous nodes will be visited by one LV. Hence, there are less than $A^{2*min(L,m)}_{2*min(L,m)}$ cases that need to be examined in a full enumeration method. Furthermore, as there are priority constraints between visiting drop-off and pick-up nodes belonging to the same SAV route, for example an LV should visit the drop-off node first before it can access the pick-up node on the same SAV route, then the possible combinations of visiting sequences will be significantly reduced.

If one of the two values L or m is small, we can connect the SAV routes belonging to one LV entirely through an exact algorithm. For the instances studied here, $L = 4, m = 3, 4, 5$. Hence, we choose a simple and adaptable backtracking algorithm to address the connection between an LV and its SAVs.

The backtracking algorithm involves a depth-first search scheme and recursive invocation. After the LV visits a rendezvous node, the collection of accessible rendezvous nodes is synchronously updated. Pruning is also implemented if the current arrival time of the LV is greater than the pre-calculated latest allowed arrival time of the LV at the rendezvous node. Note that the general constraints are checked during the whole flow of the algorithm for pruning. The algorithm records and outputs the best solution finally.

The general structure of the backtracking connection algorithm is depicted in Algorithm 4. If current accessible points set S_2 is empty (line 1), check and update the best solution (Line 2-8). Return the status $true/false$, best link S^{best} and best value V. If S_2 is not empty (line 1), execute (lines 10-22). *JudgeTimeWindow* and *UpdateCollection* procedures are run sequentially if connecting node i does not violate the *JudgeTimeWindow* constraints (lines 12-14). *Backtracking_Connection* calls itself to obtain the new S^{best} and V if its status is *true* (line 15). If V is larger than a predetermined value that the LV-SAV route value cannot reach, return status as *false* (lines 20-22).

We introduce the *UpdateCollection* and *JudgeTimeWindow* procedures as follows.

UpdateCollection: For the SAV routes belonging to one LV, there are priority constraints between the drop-off and pick-up nodes. For example, there may be several viable SAV routes starting from the same drop-off node but ending at different pick-up nodes. These pick-up nodes have to be visited after the drop-off node has been visited. Similarly, there may be several viable SAV routes ending at the same pick-up node but starting from different drop-off nodes.

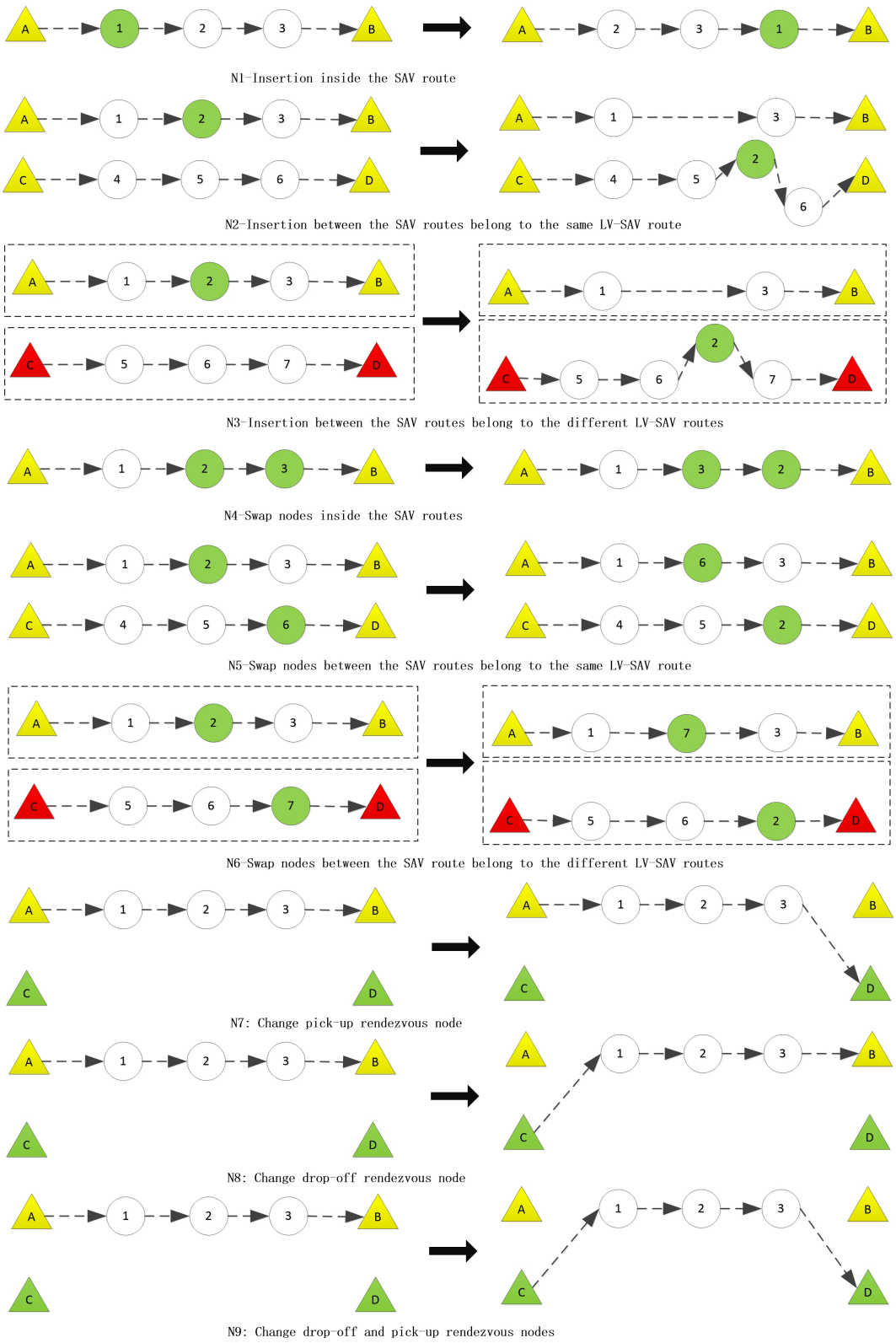

Figure 10: Local search operator

Algorithm 4 *Backtracking_Connection*$(S_1, S_2, S^{best}, T, V, P)$

if S_2 is empty **then**
  if the value of S_1 is large than V **then**
    $V \leftarrow value(S_1)$
    $S^{best} \leftarrow S_1$
    return$(true, S^{best}, V)$
  else
    return$(false, S^{best}, V)$
  end if
else
  for i to S_2 **do**
    $S_1^{temp}, S_2^{temp}, T^{temp} \leftarrow Copy(S_1, S_2, T)$
    if $!(JudgeTimeWindow(i, S_1^{temp}, T^{temp}, P))$ **then**
      $T^{temp} \leftarrow JudgeTimeWindow(i, S_1^{temp}, T^{temp}, P)$
      $(S_1^{temp}, S_2^{temp}) \leftarrow UpdateCollection(i, S_1^{temp}, S_2^{temp})$
      $(S_1^{best}, V) \leftarrow !(Backtracking\_Connection(S_1^{temp}, S_2^{temp}, S^{best}, T^{temp}, V, P))$
    else
      return$(false, S_1^{best}, V)$
    end if
  end for
  if V large than predetermined largeValue **then**
    return$(false, S_1^{best}, V)$
  end if
end if
return$(true, S_1^{best}, V)$

Note: S_1: already visited node list, S_2: current could visited node set, S^{best}: best result, T: current time, V: value. P: chosen SAV routes.

This pick-up node should be visited after all its relative drop-off nodes have been visited. During the *Backtracking_Connection* algorithm, we continually update the accessible point collection (*UpdateCollection*) synchronously after one rendezvous node has been visited.

The *UpdateCollection* procedure is addressed at each rendezvous node during the backtracking algorithm. In the beginning, only drop-off nodes are allowed to be visited. Once a drop-off node has been visited, we remove it from the accessible point collection. For the current chosen drop-off node, there may be several SAV routes departing from it and ending at one or more than one pick-up nodes. For these pick-up nodes, if there is the node whose corresponding drop-off nodes have all been visited, we put it into the accessible point collection. In addition, if the algorithm visited one pick-up node, we just remove it from the accessible point collection.

JudgeTimeWindow: In order to speed up the backtracking, we calculate the latest allowed drop-off nodes arrival times for an LV as a pruning standard. If an LV arrived at the drop-off node later than its latest allowed arrival time, the route is unfeasible. We can quickly exclude permutations that are obviously not feasible.

We calculate the latest allowed drop-off node arrival time of an SAV route from back to front. First, we calculate the latest allowed arrival time of the relative pick-up node, which equals the end of the depot time window minus the travel time between depot and pick-up node. Second, we add the latest allowed arrival time of the previous node to the travel time between the previous node and the current node. Then we compare the value obtained before to the end of the customer node time window and take the largest one. Third, we count the latest allowed arrival time of the drop-off node considering the travel time from the previous node. Note that if several SAV routes depart from one drop-off node, we choose the smallest latest allowed arrival time among them.

The specific operation process calculates the latest allowed drop-off arrival time before the *Backtracking_Connection* algorithm executes. Then we make a dictionary, including the drop-off nodes and their corresponding latest allowed arrival times, to facilitate the query during *JudgeTimeWindow* procedure.

The *JudgeTimeWindow* procedure is applied to check the feasibility of the LV visiting each rendezvous node and output the departure time from that node. If the judged node is a drop-off node, query the dictionary then see if we need to prune this node or not. If the node does not need to be pruned, calculate and output the departure time of the judged node. If the judged node is the pick-up node, judge the time window constraints for the LV and SAV routes then output the departure time for the judged nodes if feasible.

5. Computational study

We performed three types of computational experiments. First, we use CPLEX to provide benchmarks for small instances and estimate the scale of the problem that the solver can manage. Second, we evaluate the performance of the multi-start heuristic, iteration number, different moves and perturbations in the hybrid metaheuristic, and then compare the hybrid metaheuristic results against the CPLEX results to analyze its performance. Third, we implement a sensitivity analysis on the LV/SAV speed ratio to see how the related speed influences the objective of the LV-SAV model.

The mathematical programming algorithm is coded in OPL. CPLEX 12.8 is used to solve the MIP model. The hybrid metaheuristic is coded in python version 3.6.5. Both CPLEX and python are executed on an Intel(R) Core(TM) 3.6GHz processor with 32 GB memory running under Windows 10. Note that python is run with single-threading.

5.1. Instance generation

Here we borrow the instances of (Van Woensel 2018) adapted to testing the 2EVRPTW problem. The specific instance generation approach can be found in (Dellaert et al. 2018). There are 12 types of instances with 3/4/5 rendezvous nodes and 15/30/50/100 customer nodes, respectively. Each type of instance counts 20 instances that can be divided into four cases according to the different time window and demand generation methods:

- For customer i of an instance of category CA. randomly generate $20 \leq a_i \leq 260$ and $a_i \leq b_i \leq a_i + 20$ and $d_z = 10$ or 20
- For customer i of an instance of category CB. randomly generate $20 \leq a_i \leq 260$ and $a_i \leq b_i \leq a_i + 20$ and $5 \leq d_z \leq 25$
- For customer i of an instance of category CC. randomly generate $60 \leq a_i \leq 360$ and $a_i \leq b_i \leq a_i + 90$ and $d_z = 10$ or 20
- For customer i of an instance of category CD. randomly generate $60 \leq a_i \leq 360$ and $a_i \leq b_i \leq a_i + 20$ and $d_z = 10$ or 20

We decreased the number of depots in the instances, as there is only one depot rather than multi-depots in LV-SAV model. In our test setting, the first depot in the instances is always chosen. In the algorithm design verification phase, we assume that speed of the LV and SAV and cost of the LV and SAV per meter are identical. Note that we propose a speed sensitivity analysis in section 5.3.2. Here, we assume that one LV can carry 4 SAVs at most, that the capacity of one SAV equals 50, that the largest travel time for one SAV is 200, and that the time window of the depot is [0, 450].

After modifying the instances, there are several unfeasible instances for our experiments. The infeasibility mainly manifested in violating the time windows of the customer and the depot even in the most optimistic situation. In order to ensure the feasibility of our instances, we maintained the absolute value width of the time window unchanged, and then adjust the time window to just not violate the constraints.

5.2. CPLEX experiment

The total computation time for CPLEX 12.8 in each instance to minimize the primary and secondary objective is limited to 5 hours (18000s), respectively. In order to speed up the search, the upper bound of the primary objective value is obtained from the construction heuristic. We also try to compute a better lower bound of the secondary objective by column generation procedure (Desaulniers et al. 2006) within 5 hours. If the column generation procedure is not completed in 5 hours, the best of a valid lower bound each iteration generated will be chosen as the column generation lower bound. We finally choose an enhanced lower bound, a better one between CPLEX lower bound

and column generation lower bound, to assess the performance of our hybrid metaheuristics. Note that the column generation procedure is coded in DOcplex by calling CPLEX 12.8. The detailed column generation procedure is in Appendix A.

For each instance, CPLEX12.8 runs with default settings until finding an optimal solution or exhausting the predetermined maximum computation time. The computational results of all instances are presented in Table 4-5. Column 1 indicates the test instances, column 2 and column 4 show the solving time for primary and secondary objective. Column 3 gives a minimum number of LVs required to be used while the columns 5, 6, and 7 are the upper bound and the lower bound of the secondary objective and the achieved gap, respectively. Column 8 and column 9 are the enhanced lower bound got by column generation and the achieved gap between CPLEX upper bound and the enhanced lower bound.

Table 4: CPLEX results for 15 customer instances

	3-15								4-15								5-15							
	TK(s)	K	TC(s)	UB	LB	Cgap(%)	ELB	Egap(%)	TK(s)	K	TC(s)	UB	LB	Cgap(%)	ELB	Egap(%)	TK(s)	K	TC(s)	UB	LB	Cgap(%)	ELB	Egap(%)
CA1	0.5	2.0	5.0	383.9	383.9	0.0	383.9	0.0	0.6	2.0	29.0	411.3	411.3	0.0	411.3	0.0	2.2	2.0	8.8	403.7	403.7	0.0	403.7	0.0
CA2	0.7	2.0	1.1	378.5	378.5	0.0	378.5	0.0	0.5	2.0	11.7	407.8	407.8	0.0	407.8	0.0	0.5	2.0	90.3	420.5	420.4	0.0	420.4	0.0
CA3	0.7	2.0	1.9	387.8	387.8	0.0	387.8	0.0	0.6	2.0	10.9	444.7	444.7	0.0	444.7	0.0	0.8	2.0	65.7	360.2	360.2	0.0	360.2	0.0
CA4	0.4	2.0	2.6	400.9	400.9	0.0	400.9	0.0	1.4	1.0	93.7	397.8	397.8	0.0	397.8	0.0	1.7	2.0	108.1	345.9	345.9	0.0	345.9	0.0
CA5	2.6	2.0	4.5	382.8	382.8	0.0	382.8	0.0	0.4	1.0	1.5	308.7	308.7	0.0	308.7	0.0	2.3	2.0	25.5	386.7	386.7	0.0	386.7	0.0
CB1	1.7	2.0	2.2	391.8	391.7	0.0	391.8	0.0	1.3	2.0	11.3	414.5	414.4	0.0	414.5	0.0	253.0	1.0	5623.8	530.0	530.0	0.0	530.0	0.0
CB2	0.5	2.0	1.8	410.4	410.4	0.0	410.4	0.0	0.6	2.0	28.6	446.9	446.9	0.0	446.9	0.0	2.6	2.0	47.1	406.9	406.8	0.0	406.9	0.0
CB3	0.9	2.0	61.6	448.4	448.4	0.0	448.4	0.0	1.4	2.0	30.9	438.5	438.5	0.0	438.5	0.0	2.3	2.0	67.0	366.2	366.2	0.0	366.2	0.0
CB4	1.1	2.0	2.4	377.6	377.5	0.0	377.6	0.0	0.8	2.0	52.6	393.6	393.5	0.0	393.6	0.0	8.2	1.0	124.5	335.2	335.2	0.0	335.2	0.0
CB5	2.6	1.0	4.3	408.1	408.1	0.0	408.1	0.0	1.2	1.0	2.0	361.6	361.6	0.0	361.6	0.0	2.4	2.0	54.9	396.6	396.6	0.0	396.6	0.0
CC1	2064.6	2.0	31.8	361.9	361.8	0.0	361.8	0.0	1038.3	2.0	18000.0	419.9	346.4	17.5	415.5	1.0	0.6	1.0	4321.8	340.3	340.3	0.0	340.3	0.0
CC2	18000.0	2.0	182.1	344.8	344.7	0.0	344.8	0.0	770.0	2.0	530.7	391.5	391.5	0.0	391.5	0.0	2.4	1.0	18000.0	359.8	308.0	14.4	350.7	2.5
CC3	448.6	2.0	97.8	380.9	380.9	0.0	380.9	0.0	18000.0	2.0	18000.0	435.9	369.8	15.2	419.8	3.7	1764.7	2.0	18000.0	354.5	284.6	19.7	337.9	4.7
CC4	1225.0	2.0	2633.8	379.3	379.3	0.0	379.3	0.0	17.2	2.0	18000.0	377.6	326.4	13.6	364.4	3.5	18000.0	2.0	13593.3	347.3	347.2	0.0	347.2	0.0
CC5	0.4	1.0	1107.2	286.5	286.4	0.0	286.5	0.0	0.8	1.0	29.6	301.4	301.4	0.0	301.4	0.0	300.5	2.0	18000.0	392.8	320.4	18.4	374.4	4.7
CD1	1.5	2.0	1.8	370.8	370.8	0.0	370.8	0.0	0.8	2.0	21.5	408.4	408.4	0.0	408.4	0.0	1.8	2.0	9.5	385.4	385.4	0.0	385.4	0.0
CD2	0.2	1.0	1.6	348.8	348.8	0.0	348.8	0.0	2.0	2.0	6.8	396.9	396.9	0.0	396.9	0.0	0.6	1.0	6.0	384.2	384.2	0.0	384.2	0.0
CD3	0.2	1.0	1.4	382.1	382.1	0.0	382.1	0.0	1.5	2.0	33.0	451.2	451.2	0.0	451.2	0.0	11.8	2.0	50.1	351.2	351.2	0.0	351.2	0.0
CD4	0.5	2.0	6.2	381.6	381.6	0.0	381.6	0.0	1.2	2.0	18.5	355.4	355.3	0.0	355.4	0.0	4.0	2.0	42.5	339.5	339.4	0.0	339.4	0.0
CD5	0.5	2.0	2.2	368.9	368.9	0.0	368.9	0.0	0.7	1.0	29.4	301.4	301.4	0.0	301.4	0.0	1.6	2.0	13.7	404.3	404.2	0.0	404.2	0.0

Table 5: CPLEX results for 30 customer instances

	3-30								4-30								5-30							
	TK(s)	K	TC(s)	UB	LB	Cgap(%)	ELB	Egap(%)	TK(s)	K	TC(s)	UB	LB	Cgap(%)	ELB	Egap(%)	TK(s)	K	TC(s)	UB	LB	Cgap(%)	ELB	Egap(%)
CA1	54.9	3.0	18000.0	756.6	687.2	9.2	752.7	0.5	123.7	3.0	18000.0	702.3	594.0	15.4	693.7	1.2	16.0	3.0	18000.0	624.4	558.6	10.5	602.2	3.6
CA2	55.8	3.0	18000.0	653.1	618.9	5.2	653.1	0.0	19.5	3.0	18000.0	687.1	613.5	10.7	676.8	1.5	18.5	3.0	18000.0	635.7	552.4	13.1	624.8	1.7
CA3	26.6	3.0	18000.0	688.5	616.0	10.5	680.3	1.2	141.1	3.0	18000.0	649.1	556.2	14.3	639.5	1.5	71.3	3.0	2050.1	602.6	602.5	0.0	602.5	0.0
CA4	34.2	3.0	2100.4	579.5	579.4	0.0	579.4	0.0	24.2	3.0	18000.0	688.8	590.2	14.3	676.3	1.8	18.8	3.0	18000.0	662.2	578.1	12.7	578.1	12.7
CA5	30.8	3.0	18000.0	650.1	577.2	11.2	650.1	0.0	17.8	3.0	18000.0	585.1	542.5	7.3	585.1	0.0	6845.4	3.0	18000.0	649.4	488.1	24.8	488.1	24.8
CB1	192.3	3.0	18000.0	768.8	706.3	8.1	758.2	1.4	41.1	3.0	18000.0	713.6	563.7	21.0	673.1	5.7	180.9	3.0	18000.0	644.3	512.6	20.4	580.1	10.0
CB2	6.4	3.0	14885.8	632.2	632.2	0.0	632.2	0.0	161.5	3.0	18000.0	647.5	584.2	9.8	642.9	0.7	6844.3	3.0	18000.0	640.0	531.0	17.0	588.1	8.1
CB3	130.8	3.0	18000.0	691.3	606.4	12.3	679.9	1.6	46.8	3.0	18000.0	651.7	564.5	13.4	623.1	4.4	572.4	3.0	18000.0	637.8	534.5	16.2	534.5	16.2
CB4	17.0	3.0	18000.0	613.0	557.4	9.1	609.2	0.6	121.6	3.0	18000.0	689.7	561.9	18.5	654.1	5.2	18000.0	3.0	18000.0	681.6	572.2	16.1	572.2	16.1
CB5	30.4	2.0	2809.0	623.6	623.5	0.0	623.5	0.0	175.8	3.0	18000.0	615.5	553.9	10.0	601.9	2.2	18000.0	3.0	18000.0	690.5	498.2	27.8	498.2	27.8
CC1	18000.0	2.0	18000.0	658.9	436.8	33.7	436.8	33.7	18000.0	3.0	18000.0	713.4	418.0	41.4	503.3	29.5	18000.0	3.0	18000.0	685.4	407.1	40.6	407.1	40.6
CC2	18000.0	3.0	18000.0	695.8	381.1	45.2	381.1	45.2	18000.0	3.0	18000.0	647.5	466.0	28.0	466.0	28.0	18000.0	3.0	18000.0	611.9	399.4	34.7	399.4	34.7
CC3	18000.0	3.0	18000.0	651.8	336.6	48.4	450.8	30.8	18000.0	3.0	18000.0	671.3	436.5	35.0	436.5	35.0	18000.0	3.0	18000.0	670.0	460.4	31.3	460.4	31.3
CC4	18000.0	3.0	18000.0	599.9	397.9	33.7	442.0	26.3	18000.0	3.0	18000.0	627.5	440.9	29.7	453.8	27.7	18000.0	3.0	18000.0	682.1	431.6	36.7	431.6	36.7
CC5	18000.0	3.0	18000.0	651.0	381.7	41.4	432.7	33.5	18000.0	3.0	18000.0	595.3	418.3	29.7	418.3	29.7	18000.0	2.0	18000.0	628.4	340.6	45.8	340.6	45.8
CD1	47.4	2.0	977.0	619.7	619.7	0.0	619.7	0.0	17.8	3.0	18000.0	688.7	624.0	9.4	671.2	2.5	19.2	3.0	18000.0	656.4	522.4	20.4	624.5	4.9
CD2	31.3	3.0	15046.2	628.3	628.2	0.0	628.2	0.0	1039.3	3.0	18000.0	647.4	548.5	15.3	643.9	0.5	16.7	3.0	18000.0	635.0	538.4	15.2	612.7	3.5
CD3	315.3	3.0	12249.1	648.6	648.6	0.0	648.6	0.0	1143.2	3.0	18000.0	614.0	575.6	6.3	606.5	1.2	28.0	3.0	6316.7	590.1	590.0	0.0	590.0	0.0
CD4	31.2	3.0	516.0	583.9	583.9	0.0	583.9	0.0	69.0	3.0	18000.0	664.5	564.1	15.1	661.7	0.4	349.8	3.0	18000.0	673.0	570.1	15.3	630.1	6.4
CD5	12.4	3.0	18000.0	647.9	595.9	8.0	641.9	0.9	41.8	3.0	18000.0	640.6	539.0	15.9	630.5	1.6	312.3	3.0	18000.0	647.4	487.8	24.6	487.8	24.6

It is easy to observe from the results that CPLEX could solve most instances of 15 customers with 3, 4, and 5 satellites. 3/6 instances cannot be solved in 5 hours for the primary/secondary objective. For 30 customers instances, 17/60 instances cannot be solved within the specified time. While the secondary objective of most 30 customer instances cannot be solved within 5 hours, especially for 4-30 and 5-30 satellites-customer cases. In general, data suggest that the increase in the number of satellites increases the difficulty of the problem for the primary and secondary objective.

In addition, The Egap is very lower for most 15 customers instances. For instances with larger Cgap, The value of Egap decreased significantly compared with the value of Cgap. Such as instance CC1-4-15, CC3-4-15, CC4-4-15, CC2-5-15, CC3-5-15, and CC5-5-15. For 30 customers instances, the value of Egap is lower than that of the Cgap for a lot of instances. In other words, the column generation lower bound is better than the CPLEX lower bound, and the gap improvement is significant for some instances. However, there still are instances with big Egap, which also reflects the difficulty of solving the problem.

5.3. Hybrid metaheuristic experiment

We tested 60 instances of 15 customer nodes with 3/4/5 rendezvous nodes. The hybrid metaheuristic results were compared against the CPLEX results to evaluate the multi-start heuristic, iteration number, different moves, and perturbations performances in the hybrid metaheuristic. After preliminary experiments, we selected an initial set of algorithmic components to investigate the contribution of the different algorithmic components. In which $IterN1_{max} = 5$, $IterN2_{max} = 10$, $IterN3_{max} = 50$ and the two-start heuristic contains the *optimal* and *nearest* drop-off nodes selection strategy in Algorithm 3 are chosen. Furthermore, the four moves involve swap, insertion, 2-opt, and change-satellites, as well as the two perturbations *Change-Multisatellites*, *Destroy-Repair-Reconstruction*, are used.

In Tables 6-9, the AT(s) row is the average runtime for the hybrid metaheuristic, the gap0(%) row represents the gap between the average hybrid metaheuristic result and the CPLEX upper bound (baseline) for the secondary objective value. The average gap0 and runtime were obtained by solving each instance 10 times.

Based on the performance evaluation of the different algorithmic components, we designed an efficient hybrid metaheuristic. We compared the results obtained using the hybrid metaheuristic against the results with CPLEX to see how the algorithm performs: See Section 5.3.2.

5.3.1. Multi-start, iteration number, moves and perturbations performance

First, we investigated whether the multi-start procedure performs better than the one-start procedure. Section 4.1 provided optimal and nearest drop-off node selection strategies for construction heuristics. Here we evaluate the performance of one-start (optimal/nearest) and two-start strategies at a fixed runtime and see how they perform. Note that the one-start or two-start strategy are both adopted in the construction heuristic and reconstruction.

The two-start and one-start strategies were compared with the runtime of the hybrid metaheuristic set to 70s. The results are shown in Table 6. The table quickly shows that the gap0 of the two-start strategy at gap0 1.655% is better than both one-start strategies with gap0 1.781% and gap0 2.143%. We suspect that in cases where the runtime is pre-determined, presetting two distinct initial solutions make the search space larger. Note that the drop-off node selection strategies used in the multi-start procedure are also involved in the reconstruction phase of perturbation.

Table 6: Comparison of multi-start and one-start strategies

	Optimal-Nearest	Optimal	Nearest
gap0(%)	1.655	1.781	2.143

Second, we investigated how the quality of the solution changes as number of $IterN3_{max}$ increases. The experimental results are reported in Table 7, where row 1 represents the iteration number from 25 to 400. It is easy to observe from Table 7 that as the iteration number increases, the gap0 is trends downward while the AT is on an approximately linear upward trend. It seems that the algorithm can give a fairly good solution as long as it allowed for longer computation times.

The gap0 decreased significantly from 3.259% at iteration number 25 down to 2.050% at iteration number 50. From iteration number 50 to 200, the gap0 fell by nearly 0.9% as runtime increased from 47.186s to 185.061s. From 200 to 400 iterations, the gap0 slowly declined from 1.189% to 0.844% as runtime increased rapidly from 185.016s to 362.556s. We can choose the appropriate $IterN3_{max}$ based on the runtime limit and accuracy requirements.

Table 7: Sensitivity analysis on iteration number

	25	50	75	100	200	300	400
gap0(%)	3.259	2.050	1.732	1.527	1.189	1.031	0.844
AT(s)	25.012	47.186	69.984	94.633	185.016	277.287	362.556

Third, we compared the performances of the moves. Table 8 reports the results of our sensitivity analysis on the moves. Row 1 indicates the different combinations of moves in which the basemoves contain the swap, insertion, 2-opt, and change-satellites, and the next four columns represent the combinations of the 3 moves respectively. From the simulation results, we know that the change-satellites operator is the most time-consuming one. If we remove

the change-satellites operator, the average runtime reduces to 11.572s compared with 47.186s for the basemoves. However, without the change-satellites operator, the gap0 drops rapidly from 2.050% to 5.189%. Removing the swap or insertion operator reduces the quality of the solution to a similar extent as removing the change-satellites operator, but less time gets saved. Overall, removing swap, insertion or change-satellites increases the gap0 to over 5%, which makes the quality of the solution challenging to accept.

The 2-opt operator had almost no effect on the quality of the solution, giving a gap0 of 1.978% versus 2.050% for basemoves. Also, removing the 2-opt operation did not significantly reduce the solution time. The cause for this phenomenon may come from our instances as there were fewer than 5 customer nodes visited in each SAV route, which limited the space for the 2-opt operation. We keep this move and hope it could be useful to the problem where each SAV route contains a more significant number of customer points.

Table 8: Sensitivity analysis on the moves

	basemoves	no swap	no insertion	no 2-opt	no change-satellites
gap0(%)	2.050	6.315	5.430	1.978	5.189
AT(s)	47.186	34.167	41.007	46.843	11.572

Fourth, we analyzed the quality of each perturbation. Table 9 reports our sensitivity analysis on perturbations. Experiments kept the $IterN3_{max} = 50$ unchanged then employed the different perturbations combinations shown in row 1. CMS is for *Change-Multisatellites*, CR is for *Change-SAVRoute* and DRR if for *Destroy-Repair-Reconstruction*.

Table 9 shows that DRR has the best gap0 compared to CMS and CR. CR performed the worst but also had the shortest runtime. We also found that using the CMS-DRR combination gave the best quality of solution, with a 2.050% gap0 among all combinations. Compared to several other well-behaved perturbation combinations, like CMS-CR-DRR with 2.523% gap0 or DRR with 2.668% gap0, CMS-DRR gave better performance at a comparably similar runtime. We thus removed the CR perturbation from the final hybrid metaheuristic.

Table 9: Sensitivity analysis on the perturbations

	CMS-CR-DRR	CMS-CR	CR-DRR	CMS-DRR	CMS	CR	DRR
gap0(%)	2.523	4.766	2.937	2.050	5.832	8.755	2.668
AT(s)	44.065	39.645	39.262	47.186	46.625	28.959	50.842

5.3.2. Hybrid metaheuristic results

The hybrid metaheuristic was run for all the instances of 15/30/50/100 customer nodes. We set $IterN3_{max} = 200$ for instances with 15/30/50 customers as it guarantees good-quality solutions at acceptable runtimes. For instances with 100 customers, we set $IterN3_{max} = 50$ to ensure that the average runtime of the hardest instances with 5 satellites and 100 customers can be solved in 1 hour.

The 120 instances with 15/30 customers were compared to the results obtained by CPLEX. We found that the hybrid metaheuristic has an acceptable runtime and always reaches optimal solutions (when known) computed using CPLEX.

Tables 10-13 report the results for the hybrid metaheuristic. BK and AK are the best and average primary objective value. BC and AC are the best and average secondary objective value. SDK and SDC represent the standard deviation of the primary and secondary objective value. AT(s) is the average runtime of the hybrid metaheuristic. Tables 10-11 also feature gap1(%) and gap2(%), which are the gap between best hybrid metaheuristic result and CPLEX upper bound (baseline) and the gap between average hybrid metaheuristic result and CPLEX upper bound (baseline) for secondary objective value, respectively. Besides, the gap3(%) and gap4(%) are the gap between best hybrid metaheuristic result and enhanced lower bound (baseline) and the gap between average hybrid metaheuristic result and enhanced lower bound (baseline) for secondary objective value, respectively.

For the instances CB5-3-15, CB5-4-15, and CB1-5-15, the best/average primary objective values obtained from the hybrid metaheuristic are different from the CPLEX values, and so this data was eliminated when calculating the

average gap1, gap2, gap3, and gap4. For clarity, we put a star behind the value where there is a difference between CPLEX and our algorithm in the primary objective. All gap1, gap2, gap3, and gap4 values ≤ 0 are highlighted in bold (except those with a star). In addition, the instances with Egap(%) larger than 10% in Tables 4-5 are highlighted in italics in Tables 10-11.

First, we compared the hybrid metaheuristic results of small and medium-sized instances to CPLEX results. Tables 10-11 report comparative results. For the primary objective value, the hybrid metaheuristic performs almost as well as the CPLEX results (in Tables 4-5). Two CPLEX results are better (CB5-3-15 and CB1-5-15), while other values of the 118 instances are the same. Note that for these two instances with different primary objective values, we will eliminate them when comparing the secondary objective value since such comparisons do not make sense. For why the better secondary objective solutions are derived by our hybrid metaheuristic for these two instances, we think increasing the number of LVs used may lead the total travel distances of the LV-SAV model decrease in some cases.

Table 10: Hybrid metaheuristic for 15 customer instances (Iteration 200)

	3-15											4-15											5-15										
	BK	BC	AK	AC	SDK	SDC	AT	gap1	gap2	gap3	gap4	BK	BC	AK	AC	SDK	SDC	AT	gap1	gap2	gap3	gap4	BK	BC	AK	AC	SDK	SDC	AT	gap1	gap2	gap3	gap4
CA1	2.0	383.9	2.0	383.9	0.0	0.1	79.7	**0.0**	**0.0**	**0.0**	**0.0**	2.0	411.3	2.0	413.0	0.0	1.9	126.6	**0.0**	0.4	**0.0**	0.4	2.0	403.7	2.0	405.8	0.0	1.8	202.9	**0.0**	0.5	**0.0**	0.5
CA2	2.0	378.5	2.0	378.5	0.0	0.0	125.3	**0.0**	**0.0**	**0.0**	**0.0**	2.0	408.1	2.0	415.1	0.0	4.9	197.5	0.1	1.7	0.1	1.8	2.0	420.5	2.0	420.5	0.0	0.2	314.1	**0.0**	**0.0**	**0.0**	**0.0**
CA3	2.0	390.8	2.0	391.4	0.0	1.2	54.7	0.8	0.9	0.8	0.9	2.0	446.8	2.0	447.7	0.0	1.0	81.5	0.5	0.7	0.5	0.7	2.0	360.2	2.0	360.4	0.0	0.3	94.9	**0.0**	0.1	**0.0**	0.1
CA4	2.0	400.9	2.0	407.5	0.0	7.8	75.3	**0.0**	1.6	**0.0**	1.6	1.0	397.8	1.0	399.9	0.0	1.7	107.2	**0.0**	0.5	**0.0**	0.5	2.0	346.8	2.0	347.6	0.0	0.4	99.6	0.3	0.5	0.3	0.5
CA5	2.0	382.8	2.0	383.1	0.0	0.4	53.7	**0.0**	0.1	**0.0**	0.1	1.0	309.8	1.0	311.8	0.0	2.4	56.6	0.4	1.0	0.4	1.0	2.0	386.7	2.0	386.7	0.0	0.1	158.7	**0.0**	**0.0**	**0.0**	**0.0**
CB1	2.0	391.8	2.0	392.2	0.0	0.2	75.3	**0.0**	0.1	**0.0**	0.1	2.0	414.5	2.0	424.2	0.0	14.6	178.5	**0.0**	2.3	**0.0**	2.3	2.0*	386.2	2.0*	400.0	0.0	9.3	106.2	-37.2	-32.5	-37.2	-32.5
CB2	2.0	410.4	2.0	422.1	0.0	6.2	53.5	**0.0**	2.8	**0.0**	2.8	2.0	446.9	2.0	450.8	0.0	7.6	140.9	**0.0**	0.8	**0.0**	0.8	2.0	406.9	2.0	414.6	0.0	5.6	170.9	**0.0**	1.9	**0.0**	1.9
CB3	2.0	448.4	2.0	454.1	0.0	3.4	76.7	**0.0**	1.2	**0.0**	1.3	2.0	438.5	2.0	453.2	0.0	10.9	198.4	**0.0**	3.2	**0.0**	3.2	2.0	366.2	2.0	366.2	0.0	0.0	72.5	**0.0**	**0.0**	**0.0**	**0.0**
CB4	2.0	377.6	2.0	384.7	0.0	4.2	75.9	**0.0**	1.9	**0.0**	1.9	2.0	393.6	2.0	401.4	0.0	9.0	112.8	**0.0**	1.9	**0.0**	1.9	1.0	335.9	1.0	349.8	0.0	6.0	151.3	0.2	4.2	0.2	4.2
CB5	2.0*	355.5	2.0*	355.5	0.0	0.0	58.8	-14.8	-14.8	-14.8	-14.8	1.0	361.6	1.9*	351.8	0.3	3.5	104.5	**0.0**	-2.8	**0.0**	-2.8	2.0	396.6	2.0	398.7	0.0	2.2	201.4	**0.0**	0.5	**0.0**	0.5
CC1	2.0	361.9	2.0	362.5	0.0	1.3	150.5	**0.0**	0.2	**0.0**	0.2	2.0	419.9	2.0	425.8	0.0	3.3	360.2	**0.0**	1.4	1.0	2.4	1.0	340.3	1.0	340.3	0.0	0.0	875.5	**0.0**	**0.0**	**0.0**	**0.0**
CC2	2.0	345.8	2.0	346.9	0.0	0.4	223.8	0.3	0.6	0.3	0.6	2.0	396.1	2.0	407.0	0.0	8.4	360.2	1.2	3.8	1.2	3.8	1.0	359.8	1.0	365.2	0.0	3.0	634.3	**0.0**	1.5	2.5	4.0
CC3	2.0	380.9	2.0	390.8	0.0	5.9	133.8	**0.0**	2.5	**0.0**	2.5	2.0	435.9	2.0	448.5	0.0	6.6	340.3	**0.0**	2.8	3.7	6.4	2.0	354.5	2.0	354.5	0.0	0.0	267.5	**0.0**	**0.0**	4.7	4.7
CC4	2.0	379.3	2.0	380.7	0.0	3.4	147.2	**0.0**	0.4	**0.0**	0.4	2.0	371.6	2.0	374.1	0.0	7.9	289.8	**-1.6**	**-0.9**	1.9	2.6	2.0	348.0	2.0	349.5	0.0	0.8	474.2	0.2	0.6	0.2	0.6
CC5	1.0	286.5	1.0	286.5	0.0	0.0	150.5	**0.0**	**0.0**	**0.0**	**0.0**	1.0	301.4	1.0	308.4	0.0	3.2	253.9	**0.0**	2.3	**0.0**	2.3	2.0	392.8	2.0	397.2	0.0	4.7	541.3	**0.0**	1.1	4.7	5.7
CD1	2.0	370.8	2.0	371.9	0.0	1.7	99.2	**0.0**	0.3	**0.0**	0.3	2.0	408.4	2.0	408.8	0.0	1.3	213.5	**0.0**	0.1	**0.0**	0.1	2.0	385.4	2.0	392.6	0.0	11.1	319.1	**0.0**	1.8	**0.0**	1.8
CD2	1.0	348.8	1.0	362.3	0.0	9.3	68.1	**0.0**	3.7	**0.0**	3.7	2.0	402.3	2.0	404.3	0.0	2.6	148.4	1.3	1.8	1.3	1.8	1.0	384.2	1.0	387.6	0.0	4.8	217.4	**0.0**	0.9	**0.0**	0.9
CD3	1.0	386.9	1.0	398.9	0.0	7.7	49.5	1.2	4.2	1.2	4.2	2.0	451.2	2.0	452.7	0.0	1.6	166.6	**0.0**	0.3	**0.0**	0.3	2.0	351.2	2.0	351.2	0.0	0.0	160.2	**0.0**	**0.0**	**0.0**	**0.0**
CD4	2.0	381.6	2.0	385.6	0.0	8.5	110.3	**0.0**	1.0	**0.0**	1.0	2.0	355.4	2.0	355.4	0.0	0.0	187.3	**0.0**	**0.0**	**0.0**	**0.0**	2.0	339.5	2.0	342.8	0.0	1.2	163.3	**0.0**	1.0	**0.0**	1.0
CD5	2.0	368.9	2.0	369.6	0.0	2.2	57.2	**0.0**	0.2	**0.0**	0.2	1.0	301.4	1.0	308.1	0.0	2.7	246.6	**0.0**	2.2	**0.0**	2.2	2.0	404.3	2.0	406.5	0.0	2.6	275.8	**0.0**	0.6	**0.0**	0.6
Aver	1.9	376.6	1.9	380.4	0.0	3.2	95.9	0.1	1.1	0.1	1.1	1.8	393.6	1.8	398.1	0.0	4.8	193.6	0.1	1.4	0.5	1.8	1.8	373.5	1.8	376.9	0.0	2.7	275.1	0.0	0.8	0.7	1.4

Table 11: Hybrid metaheuristic for 30 customer instances (Iteration 200)

	3-30											4-30											5-30										
	BK	BC	AK	AC	SDK	SDC	AT	gap1	gap2	gap3	gap4	BK	BC	AK	AC	SDK	SDC	AT	gap1	gap2	gap3	gap4	BK	BC	AK	AC	SDK	SDC	AT	gap1	gap2	gap3	gap4
CA1	3.0	760.8	3.0	776.6	0.0	8.9	199.8	0.6	2.6	1.1	3.1	3.0	705.7	3.0	711.1	0.0	3.7	363.2	0.5	1.2	1.7	2.4	3.0	632.5	3.0	658.5	0.0	17.9	563.6	1.3	5.2	4.8	8.6
CA2	3.0	656.7	3.0	666.3	0.0	4.6	222.5	0.5	2.0	0.5	2.0	3.0	686.5	3.0	696.0	0.0	10.7	421.2	**-0.1**	1.3	1.4	2.8	3.0	640.7	3.0	651.6	0.0	10.6	583.3	0.8	2.4	2.5	4.1
CA3	3.0	688.5	3.0	697.5	0.0	5.9	267.4	**0.0**	1.3	1.2	2.5	3.0	653.2	3.0	668.9	0.0	10.6	385.9	0.6	3.0	2.1	4.4	3.0	602.6	3.0	604.8	0.0	2.3	596.6	**0.0**	0.4	**0.0**	0.4
CA4	3.0	585.6	3.0	596.6	0.0	9.4	198.6	1.1	2.9	1.1	2.9	3.0	684.8	3.0	702.3	0.0	10.7	307.0	**-0.6**	1.9	1.2	3.7	3.0	664.1	3.0	674.1	0.0	8.1	368.4	*0.3*	*1.8*	*12.9*	*14.2*
CA5	3.0	659.9	3.0	672.7	0.0	8.8	261.0	1.5	3.4	1.5	3.4	3.0	586.9	3.0	616.4	0.0	18.6	405.6	0.3	5.1	0.3	5.1	3.0	648.4	3.0	663.0	0.0	15.4	522.7	***-0.1***	*2.0*	*24.7*	*26.4*
CB1	3.0	768.8	3.0	778.0	0.0	5.0	189.9	**0.0**	1.2	1.4	2.5	3.0	720.0	3.0	726.6	0.0	8.1	389.3	0.9	1.8	6.5	7.4	3.0	646.8	3.0	660.5	0.0	8.8	1095.6	*0.4*	*2.5*	*10.3*	*12.2*
CB2	3.0	642.5	3.0	662.4	0.0	11.4	175.7	1.6	4.6	1.6	4.6	3.0	658.0	3.0	667.2	0.0	7.6	263.7	1.6	3.0	2.3	3.6	3.0	642.9	3.0	654.9	0.0	6.2	748.6	0.5	2.3	8.5	10.2
CB3	3.0	698.3	3.0	704.4	0.0	6.1	250.6	1.0	1.9	2.6	3.5	3.0	645.9	3.0	664.7	0.0	12.5	467.3	**-0.9**	2.0	3.5	6.3	3.0	630.9	3.0	640.4	0.0	6.8	489.5	***-1.1***	*0.4*	*15.3*	*16.5*
CB4	3.0	613.0	3.0	623.1	0.0	11.2	227.8	**0.0**	1.6	0.6	2.2	3.0	672.9	3.0	690.4	0.0	10.0	351.7	**-2.5**	0.1	2.8	5.3	3.0	683.5	3.0	692.2	0.0	5.8	278.3	*0.3*	*1.5*	*16.3*	*17.3*
CB5	2.0	651.1	2.0	670.4	0.0	12.4	100.4	4.2	7.0	4.2	7.0	3.0	612.6	3.0	632.8	0.0	13.2	479.3	**-0.5**	2.7	1.7	4.9	3.0	673.0	3.0	685.6	0.0	12.4	1158.0	***-2.6***	***-0.7***	*26.0*	*27.3*
CC1	2.0	615.9	2.0	631.6	0.0	14.0	234.2	***-7.0***	***-4.3***	*29.1*	*30.8*	3.0	648.8	3.0	658.2	0.0	10.6	822.2	***-10.0***	***-8.4***	*22.4*	*23.5*	3.0	623.4	3.0	635.8	0.0	16.6	3632.6	***-9.9***	***-7.8***	*34.7*	*36.0*
CC2	3.0	624.3	3.0	632.1	0.0	7.4	502.0	***-11.5***	***-10.1***	*39.0*	*39.7*	3.0	645.0	3.0	651.3	0.0	5.7	885.5	***-0.4***	*0.6*	*27.8*	*28.5*	3.0	605.9	3.0	622.0	0.0	14.6	1575.5	***-1.0***	*1.6*	*34.1*	*35.8*
CC3	3.0	640.3	3.0	650.3	0.0	10.4	579.0	***-1.8***	***-0.2***	*29.6*	*30.7*	3.0	616.1	3.0	630.3	0.0	13.6	777.5	***-8.9***	***-6.5***	*29.2*	*30.7*	3.0	624.3	3.0	636.0	0.0	8.0	1199.9	***-7.3***	***-5.3***	*26.2*	*27.6*
CC4	3.0	568.7	3.0	574.3	0.0	6.0	438.8	***-5.5***	***-4.5***	*22.3*	*23.0*	3.0	614.9	3.0	628.2	0.0	9.4	943.3	***-2.0***	*0.1*	*26.2*	*27.8*	3.0	647.4	3.0	653.4	0.0	4.2	1175.1	***-5.4***	***-4.4***	*33.3*	*34.0*
CC5	3.0	633.1	3.0	639.9	0.0	7.1	469.0	***-2.8***	***-1.7***	*31.7*	*32.4*	3.0	573.7	3.0	588.1	0.0	8.2	932.0	***-3.8***	***-1.2***	*27.1*	*28.9*	2.0	546.9	2.0	558.2	0.0	8.6	2279.1	***-14.9***	***-12.6***	*37.7*	*39.0*
CD1	2.0	632.3	2.0	670.2	0.0	18.7	133.7	2.0	7.5	2.0	7.5	3.0	692.5	3.0	699.4	0.0	7.5	553.2	0.5	1.5	3.1	4.0	3.0	651.4	3.0	661.4	0.0	9.3	2194.6	**-0.8**	0.8	4.1	5.6
CD2	3.0	636.2	3.0	645.7	0.0	7.3	332.4	1.2	2.7	1.3	2.7	3.0	646.4	3.0	661.0	0.0	11.5	516.5	**-0.2**	2.1	0.4	2.6	3.0	634.8	3.0	647.1	0.0	7.8	939.4	**0.0**	1.9	3.5	5.3
CD3	3.0	650.3	3.0	667.7	0.0	11.1	380.2	0.3	2.9	0.3	2.9	3.0	621.8	3.0	634.2	0.0	9.6	557.7	1.2	3.2	2.5	4.4	3.0	598.6	3.0	608.4	0.0	9.6	1021.7	1.4	3.0	1.4	3.0
CD4	3.0	583.9	3.0	597.7	0.0	12.7	367.3	**0.0**	2.3	**0.0**	2.3	3.0	667.2	3.0	684.0	0.0	18.5	567.2	0.4	2.9	0.8	3.3	3.0	666.8	3.0	673.2	0.0	2.9	584.0	**-0.9**	**0.0**	5.5	6.4
CD5	3.0	648.4	3.0	661.3	0.0	13.3	338.6	0.1	2.0	1.0	2.9	3.0	637.0	3.0	645.6	0.0	8.7	783.8	**-0.6**	0.8	1.0	2.4	3.0	621.9	3.0	645.6	0.0	11.9	928.6	***-4.1***	***-0.3***	*21.6*	*24.4*
Aver	2.9	647.9	2.9	660.9	0.0	9.6	293.4	-0.7	1.2	8.6	10.4	3.0	649.5	3.0	662.8	0.0	10.4	558.7	-1.2	0.9	8.2	10.1	3.0	634.3	3.0	646.3	0.0	9.4	1096.8	-2.2	-0.3	16.2	17.7

For the secondary objective value, more than two thirds of the hybrid metaheuristic's best results are the same as or even better than the CPLEX upper bounds (gap1 $\leq$ 0). Furthermore, the average gap2 is below 0.9% for 15/30 customers instances. Note that for the instances with 5 satellites and 30 customers, the hybrid metaheuristic performs much better than CPLEX upper bound.

Besides, more than two-thirds of the hybrid metaheuristic's best results are the same as the enhanced lower bound (gap3 = 0), and the average gap4 is nearly 1.43% for 15 customers instances. For 30 customers instances, some best results of hybrid metaheuristic have a larger gap with the enhanced lower bounds. The main reason may be the enhanced lower bounds we got are still weak, and the enhanced lower bound may have a larger gap with the optimal value. If we remove the instances of 30 customers with Egap > 10% (highlighted in italics in Tables 11), the average gap3 and gap4 are acceptable, 2.2% and 4.2% respectively.

The results for 50 and 100 customers are shown in Tables 12-13, which show that as the number of rendezvous nodes increases, the average secondary objective value trends downward. Hence, a reasonable increase in rendezvous nodes is useful for saving transportation costs.

For all the instances, category CC instances with wide time windows had the longest runtimes. We suspect that the wider time windows result in a larger solution space, and our hybrid metaheuristic contains backtracking algorithms that need an exact search of the solution space, so category CC instances cause long runtimes.

Table 12: Hybrid metaheuristic for 50 customer instances (Iteration 200)

	3-50							4-50							5-50						
	BK	BC	AK	AC	SDK	SDC	AT(S)	BK	BC	AK	AC	SDK	SDC	AT(S)	BK	BC	AK	AC	SDK	SDC	AT(S)
CA1	4.0	1029.5	4.0	1057.6	0.0	19.4	587.4	4.0	965.6	4.0	984.3	0.0	12.3	1214.1	4.0	926.5	4.0	961.7	0.0	35.7	1611.3
CA2	4.0	1066.8	4.0	1076.7	0.0	8.8	774.4	4.0	983.2	4.0	996.5	0.0	10.8	1959.4	4.0	913.7	4.0	930.3	0.0	11.6	1276.4
CA3	4.0	1012.7	4.0	1034.7	0.0	17.2	1064.6	4.0	969.6	4.0	982.9	0.0	7.2	1178.8	4.0	889.9	4.0	904.2	0.0	9.8	2520.1
CA4	4.0	993.2	4.0	1000.2	0.0	5.8	1028.6	4.0	840.5	4.0	854.8	0.0	13.1	1395.4	4.0	920.6	4.0	935.9	0.0	10.8	2452.0
CA5	4.0	952.4	4.0	964.3	0.0	7.5	881.8	4.0	943.8	4.0	964.1	0.0	21.7	1382.3	4.0	927.3	4.0	951.8	0.0	15.9	2609.6
CB1	4.0	994.0	4.0	1007.1	0.0	10.1	1068.1	4.0	965.7	4.0	1004.3	0.0	17.4	940.4	4.0	922.5	4.0	945.7	0.0	21.4	1451.0
CB2	4.0	1000.0	4.0	1016.0	0.0	12.6	791.2	4.0	965.7	4.0	1004.3	0.0	17.4	940.4	4.0	971.8	4.0	1006.9	0.0	28.0	952.8
CB3	4.0	1022.4	4.0	1053.9	0.0	23.1	742.0	4.0	986.6	4.0	1024.8	0.0	30.3	1114.2	4.0	1047.0	4.9	1044.2	0.3	11.8	1765.6
CB4	4.0	983.8	4.0	994.8	0.0	8.6	714.9	4.0	888.4	4.0	919.7	0.0	18.6	1060.1	4.0	896.1	4.0	945.5	0.0	29.2	1586.6
CB5	4.0	982.1	4.0	1003.6	0.0	13.7	1056.8	4.0	994.0	4.0	1020.0	0.0	19.3	1313.1	4.0	908.8	4.0	935.0	0.0	14.9	2689.2
CC1	4.0	940.8	4.0	946.6	0.0	5.2	2384.6	4.0	866.5	4.0	879.6	0.0	10.6	3386.5	4.0	853.7	4.0	876.3	0.0	14.0	5573.0
CC2	4.0	990.4	4.0	1003.5	0.0	12.1	2017.8	4.0	886.3	4.0	909.4	0.0	12.9	3278.9	4.0	847.2	4.0	879.2	0.0	23.3	7529.1
CC3	4.0	902.6	4.0	920.6	0.0	11.0	2071.6	4.0	898.4	4.0	909.5	0.0	9.0	3246.4	4.0	876.3	4.0	886.9	0.0	5.1	4812.3
CC4	4.0	941.4	4.0	947.3	0.0	4.8	1796.6	4.0	792.4	4.0	807.2	0.0	11.6	3420.2	4.0	901.6	4.2	920.9	0.4	17.0	3594.1
CC5	4.0	1001.6	4.0	1008.9	0.0	5.4	1861.8	5.0	996.9	5.0	1021.4	0.0	16.8	3449.4	4.0	889.2	4.0	911.3	0.0	23.5	5779.2
CD1	4.0	993.9	4.0	1006.3	0.0	11.0	1580.8	4.0	913.8	4.0	935.0	0.0	13.4	2391.0	4.0	871.3	4.0	889.8	0.0	13.5	3628.7
CD2	4.0	1067.8	4.0	1080.9	0.0	10.8	1029.8	4.0	943.4	4.0	951.7	0.0	6.4	1867.2	4.0	873.6	4.0	886.0	0.0	9.6	4995.0
CD3	4.0	960.3	4.0	980.2	0.0	17.9	1352.9	4.0	988.6	4.0	1007.8	0.0	14.3	1448.6	4.0	910.3	4.0	931.4	0.0	10.8	3166.7
CD4	4.0	977.4	4.0	993.9	0.0	13.3	1371.3	4.0	822.5	4.0	845.6	0.0	15.8	2018.1	4.0	863.7	4.0	889.8	0.0	20.1	3783.1
CD5	4.0	1011.0	4.0	1042.4	0.0	27.1	818.3	4.0	968.2	4.0	986.0	0.0	11.8	2091.0	4.0	901.9	4.0	926.0	0.0	14.5	3909.1
Aver	4.0	991.2	4.0	1007.0	0.0	12.3	1249.8	4.1	929.0	4.1	950.4	0.0	14.5	1954.8	4.0	905.6	4.1	927.9	0.0	17.0	3284.3

Table 13: Hybrid metaheuristic for 100 customer instances (Iteration 50)

	3-100							4-100							5-100						
	BK	BC	AK	AC	SDK	SDC	AT(S)	BK	BC	AK	AC	SDK	SDC	AT(S)	BK	BC	AK	AC	SDK	SDC	AT(S)
CA1	8.0	1925.3	8.0	1977.6	0.0	29.6	1234.0	8.0	1891.8	8.0	1927.7	0.0	29.9	1485.4	8.0	1812.6	8.0	1855.4	0.0	34.9	2296.4
CA2	8.0	1958.4	8.0	2033.9	0.0	61.1	1046.9	7.0	1810.2	7.0	1864.2	0.0	68.3	886.5	8.0	1699.8	8.0	1758.0	0.0	34.7	2018.1
CA3	8.0	1990.1	8.0	2035.5	0.0	29.8	1111.3	8.0	2069.1	8.0	2142.5	0.0	45.8	1377.7	8.0	1705.4	8.0	1777.1	0.0	39.1	1657.8
CA4	8.0	2315.4	8.0	2380.8	0.0	38.2	1077.1	7.0	1700.8	7.0	1764.3	0.0	35.1	1317.1	8.0	1798.1	8.0	1868.6	0.0	40.1	2266.2
CA5	8.0	1855.0	8.0	1916.0	0.0	33.6	1669.4	8.0	1817.0	8.0	1867.0	0.0	35.8	2476.8	8.0	1761.4	8.0	1842.0	0.0	46.3	2213.5
CB1	7.0	1852.0	7.0	1924.5	0.0	43.8	839.4	7.0	1899.4	7.0	1986.6	0.0	61.0	774.4	8.0	1823.6	8.0	1904.3	0.0	50.5	2129.1
CB2	8.0	1964.6	8.0	2027.9	0.0	45.2	1040.4	8.0	1892.5	8.0	1929.6	0.0	27.9	1607.0	8.0	1711.7	8.0	1797.2	0.0	43.5	1886.0
CB3	8.0	1970.2	8.0	2026.8	0.0	32.7	1156.2	8.0	2090.3	8.0	2158.7	0.0	40.6	1318.7	8.0	1767.4	8.0	1847.4	0.0	35.8	1854.0
CB4	8.0	2301.5	8.0	2365.4	0.0	38.2	828.2	8.0	2122.7	8.9	2058.1	0.3	42.4	1385.4	8.0	1960.6	8.4	2019.5	0.5	65.0	1466.6
CB5	8.0	1902.0	8.0	1941.2	0.0	27.0	1181.1	8.0	1890.6	8.0	1976.2	0.0	47.2	1144.2	8.0	1910.7	8.0	2047.3	0.0	98.5	1294.1
CC1	8.0	1847.9	8.0	1891.1	0.0	28.7	3557.8	8.0	1737.3	8.0	1813.6	0.0	38.1	3153.3	8.0	1695.2	8.0	1764.4	0.0	42.7	6271.8
CC2	8.0	1742.4	8.0	1780.9	0.0	30.2	2483.4	8.0	1771.7	8.0	1828.4	0.0	33.1	4161.9	8.0	1600.9	8.0	1641.2	0.0	31.6	4970.2
CC3	8.0	1875.0	8.0	1914.0	0.0	35.1	1994.1	7.0	1768.1	7.0	1849.7	0.0	37.0	5149.8	8.0	1677.1	8.0	1721.8	0.0	28.7	5113.4
CC4	8.0	2185.3	8.0	2237.0	0.0	31.4	2139.8	8.0	1741.4	8.0	1786.8	0.0	29.3	4083.3	8.0	1653.7	8.0	1728.1	0.0	35.9	5625.3
CC5	8.0	1810.1	8.0	1839.2	0.0	24.7	2439.4	8.0	1774.6	8.0	1822.2	0.0	27.7	3436.2	8.0	1720.4	8.0	1791.0	0.0	36.4	4613.2
CD1	8.0	1801.2	8.0	1846.5	0.0	35.6	1615.9	8.0	1885.6	8.0	1917.3	0.0	14.4	2173.0	8.0	3282.0	8.0	3347.5	0.0	52.6	1966.1
CD2	8.0	1927.9	8.1	2017.9	0.3	48.7	905.9	8.0	1767.5	8.0	1858.9	0.0	39.0	1825.0	8.0	1650.3	8.0	1696.1	0.0	38.0	3638.4
CD3	8.0	1931.2	8.0	1964.5	0.0	26.1	976.0	7.0	1851.5	7.0	1919.9	0.0	36.8	1673.8	8.0	1700.5	8.0	1762.6	0.0	50.3	3604.8
CD4	8.0	2201.7	8.0	2261.1	0.0	40.1	1019.2	8.0	1838.5	8.0	1905.2	0.0	33.7	1972.8	8.0	1742.0	8.0	1796.1	0.0	40.7	3604.1
CD5	8.0	1832.0	8.0	1901.4	0.0	48.8	1529.0	8.0	1827.6	8.0	1868.9	0.0	23.9	1841.5	8.0	1777.9	8.0	1841.2	0.0	38.2	3808.2
Aver	8.0	1959.5	8.0	2014.2	0.0	36.4	1492.2	7.8	1857.4	7.8	1912.3	0.0	37.4	2162.2	8.0	1822.6	8.0	1890.3	0.0	44.2	3114.9

5.4. Sensitivity analysis on related speed

As speed of the vehicles may affect the model-computed outputs, we analyzed the impact of vehicle speed changes on the results.

The speed of SAVs is commonly around walking speed (5km/h) in real-world applications. For the SAV, a speed faster than 10km/h seems to be unrealistic in pedestrianized and high density urban areas. Hence, we assume the speed of the SAV to be between 5 and 10 km/h in our experiments. Besides, the LV speed can be assumed to be usually less than 30km/h as imposed by speed limits in city centers. We assume the speed of LVs to be between 15 and 25 km/h in our experiments.

We chose simple instances to test different LV/SAV speed combinations by CPLEX to see how vehicle speeds influence the results. We set 5km/h as the baseline speed.

For the CPLEX experiments, we generated the different LV/SAV speed combinations via the following step. First, we set the speed of SAV equals to 5km/h. Then, we kept SAV speed unchanged but gradually increased LV speed from 15km/h, 20km/h, up to 25km/h. Next, we kept LV speed unchanged but increased SAV speed from 5km/h up to 10km/h.

We chose 15 simple instances with 3 satellites and 15 customers to implement the CPLEX experiments. Tables 14 shows the instances used in column 1. For the primary objective value, all results for LV/SAV speed combinations were identical. The secondary objective results are reported in Table 14, where row 1 gives the different combinations of the LV/SAV speed. Bold-type shows that results are different from the previous column.

Table 14: Sensitivity analysis on related speed by CPLEX

3-15	(15km/h,5km/h)	(20km/h,5km/h)	(25km/h,5km/h)	(25km/h,10km/h)
CA1	383.86	383.86	383.86	**383.49**
CA2	378.54	378.54	378.54	378.54
CA3	385.91	385.91	385.91	385.91
CA4	400.92	400.92	400.92	400.92
CA5	382.76	382.76	382.76	**380.72**
CB1	391.76	391.76	391.76	391.76
CB2	363.70	363.70	363.70	**351.87**
CB3	412.51	412.51	412.51	**399.63**
CB4	377.55	377.55	377.55	377.55
CB5	408.05	408.05	408.05	**353.84**
CD1	370.84	370.84	370.84	**359.50**
CD2	348.82	348.82	348.82	348.82
CD3	382.10	382.10	382.10	382.10
CD4	381.56	381.56	381.56	381.56
CD5	368.87	368.87	368.87	**352.81**
Aver	382.52	382.52	382.52	**375.27**

In experiments with simple instances, we found that increasing the LV speed from 15km/h to 25km/h has little effect on reducing the secondary objective function. However, increasing the SAV speed helps reduce average costs. For example, increasing SAV speed from 5km/h to 10km/h reduces the secondary objective value from 382.52 to 375.27. We assumed that the bottleneck of the LV-SAV delivery system may be caused by the speed of the SAV.

Hence, we implemented an in-depth experimental analysis on the speed of SAV. We conducted the hybrid meta-heuristic to test all instances with 15 and 30 customers. The LV/SAV speed combination (25km/h,5km/h) was chosen as the control group. We fixed the speed of the LV and let the speed of the SAVs be 6.25km/h, 7.5km/h,..., 10km/h, as the experimental groups. There are 120 instances, and every instance is run 10 times with *Iter3=50* for each type of LV/SAV speed combination.

For the primary objective, the best results in the experimental group are all equal to the control group results in all the experiments. This comes from two factors: First, the number of LVs used depends on a variety of constraints, not only the speed of the vehicles but also on vehicle capacity constraints, maximum travel time constraints, and so on. As long as one type of constraint is limiting, the number of LVs used cannot be reduced. Second, in our instances, the value of the primary objective is usually 1,2 or 3. The need to cause changes in vehicle numbers often requires a significant change in conditions.

Table 15 shows the sensitivity analysis on the secondary objective. Row 1 represents the different combinations of LV/SAV speed. Row 2 and row 3 are the average best results for the 15 customers and 30 customers instances, in which the control group result (baseline) is the average best secondary objective value and the quality of the experimental group solution is reported as an average percentage gap from the baseline.

Table 15 shows that as SAV speed grows, the quality of solutions for the secondary objective value increases. However, the total improvement is limited. Increasing the speed of the SAV from 5km/h to 10km/h reduces the cost by less than 0.6% of the cost for our 15 customers and 30 customers instances. We suspect that the improvement of the objective value is affected by many factors (constraints). Simply improving the speed of SAV can improve the solution, but improvements remains limited.

We therefore recommend to keep SAV speeds rather low because of increased safety and a more convenient environment for pedestrians in practical implementations.

Table 15: Sensitivity analysis on speed with the hybrid metaheuristic

	baseline(25km/h,5km/h)	(25km/h,6.25km/h)	(25km/h,7.5km/h)	(25km/h,8.75km/h)	(25km/h,10km/h)
15 cust best cost	380.13	-0.23%	-0.34%	-0.50%	-0.58%
30 cust best cost	648.20	-0.21%	-0.42%	-0.50%	-0.50%

6. Conclusion and future work

This paper provides an efficient transportation delivery model for the logistics distribution of autonomous vehicles in the city. We investigate an innovative two-echelon urban delivery problem using autonomous vehicles that contains the time window constraint, capacity constraint and maximum travel time constraint. The LV carries the SAVs on the *1st-level route* and drops off and picks up them in the rendezvous nodes, while the SAV handles customer service on the *2nd-level route*. In addition, our model allows each LV carries multiple SAVs, and each SAV can visit multiple customers during a trip. This innovation distribution model can eliminate a lot of real estates and manpower compared to the traditional two-echelon delivery model. Which provides a new choice for logistics enterprises.

We first define this problem and present its mixed linear integer programming formulation, and present a column generation procedure to try to get a better lower bound. Since CPLEX cannot solve medium-sized and large-scale problems, we introduce construction heuristics that can provide a nice upper bound for the CPLEX solver and quickly generate the feasible initial solutions. The construction heuristic first uses a modified nearest neighbor approach to construct multiple SAV routes (*1st-level route*), then adopts a simple connection heuristic to construct multiple LV routes (*1st-level route*) to get the final solution. In addition, a hybrid metaheuristic that involves multi-start, destroy, repair, iterative local search, and backtracking is presented to quickly solve the medium-sized and large-scale problems that CPLEX cannot address. And a performance evaluation of the different algorithmic components for hybrid metaheuristic are implemented. Besides, we used the CPLEX upper bound and enhanced lower bound to assess the hybrid metaheuristic for the small-sized problem. Assessment results show that our hybrid metaheuristic is competitive. Furthermore, we have provided a benchmark solution for the problem along with a sensitivity analysis for the LV/SAV speed combinations. In general, increasing the speed of the SAV rather than the speed of LV can reduce the objective function value in real-word applications. However, simply improving the speed of the SAV leads to only limited reduction in cost. We therefore recommend to keep SAV speeds rather low because of a more pedestrian friendly environment in practical implementations.

Future research could go beyond some of the simplifying assumptions and extend the operational model of our approach. For example, we could allow the SAVs dropped off by an LV to be picked up by a different LV, rather than having to be picked up by the same one. Removing this assumption might lead to more cost-efficient solutions. We could also allow LVs to carry SAVs and freight, and further performing freight replenishment during LV routes. Besides, other cost functions of the LV-SAV model, like travel time, energy consumption, and carbon emissions (Murray and Chu 2015, Jemai and Sarkar 2019, Sarkar et al. 2016), can also be considered. In addition, we noticed that there is a period when the LV carries its SAVs moving on the road. If we could use this period to conduct meaningful en-route operations, the efficiency of the LV-SAV model might be improved. Hence, en-route operations in the LV-SAV model could be a promising research direction. Moreover, we could assume the LV and SAV are electric vehicles. The LV and SAV can be charged at charging stations, and the SAV also can be charged by the LV en-route. This new charging model may reduce the extra time required for charging in the distribution system. It could improve the availability of electric vehicles for urban delivery, and thus further reduce local emissions in cities.

7. Acknowledgements

This work was supported by the China Scholarship Council and by public funding within the scope of the French Program "Investissements d'Avenir".

References

Agatz, N., Bouman, P., and Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, 52(4):965–981.

Andrew, J. H. (2019). Thousands of autonomous delivery robots are about to descend on us college campuses. https://www.theverge.com/2019/8/20/20812184/starship-delivery-robot-expansion-college-campus. Accessed August 20, 2019.

Bala, K., Brcanov, D., and Gvozdenović, N. (2017). Two-echelon location routing synchronized with production schedules and time windows. *Central European Journal of Operations Research*, 25(3):525–543.

Baldacci, R., Mingozzi, A., Roberti, R., and Calvo, R. W. (2013). An exact algorithm for the two-echelon capacitated vehicle routing problem. *Operations Research*, 61(2):298–314.

Bouman, P., Agatz, N., and Schmidt, M. (2018). Dynamic programming approaches for the traveling salesman problem with drone. *Networks*, 72(4):528–542.

Boysen, N., Schwerdfeger, S., and Weidinger, F. (2018). Scheduling last-mile deliveries with truck-based autonomous robots. *European Journal of Operational Research*, 271(3):1085–1099.

Braekers, K., Ramaekers, K., and Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313.

Brandão, J. (2018). Iterated local search algorithm with ejection chains for the open vehicle routing problem with time windows. *Computers & Industrial Engineering*, 120:146–159.

Breunig, U., Schmid, V., Hartl, R. F., and Vidal, T. (2016). A large neighbourhood based heuristic for two-echelon routing problems. *Computers & Operations Research*, 76:208–225.

Carlsson, J. G. and Song, S. (2017). Coordinated logistics with a truck and a drone. *Management Science*, 64(9):4052–4069.

Chao, I.-M. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29(1):33–51.

Chevallier, H. (2017). Twinswheel, le livreur du futur. https://www.franceinter.fr/emissions/c-est-deja-demain/c-est-deja-demain-29-novembre-2017. Accessed November 29, 2017.

Crainic, T. G., Sforza, A., and Sterle, C. (2011). *Location-routing models for two-echelon freight distribution system design*. CIRRELT.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812.

Cuda, R., Guastaroba, G., and Speranza, M. G. (2015). A survey on two-echelon routing problems. *Computers & Operations Research*, 55:185–199.

Daimler (2017). Vans and robots paketbote 2.0. https://www.daimler.com/innovation/specials/future-transportation-vans/paketbote-2-0.html. Accessed Dec 25, 2018.

Dellaert, N., Dashty Saridarq, F., Van Woensel, T., and Crainic, T. G. (2018). Branch-and-price–based algorithms for the two-echelon vehicle routing problem with time windows. *Transportation Science*, 53(2):463–479.

Desaulniers, G., Desrosiers, J., and Solomon, M. M. (2006). *Column generation*, volume 5. Springer Science & Business Media.

Desrosiers, J. and Lübbecke, M. E. (2005). A primer in column generation. In *Column generation*, pages 1–32. Springer.

Duarte, A., Sánchez-Oro, J., Mladenović, N., and Todosijević, R. (2018). Variable neighborhood descent. *Handbook of Heuristics*, pages 341–367.

European Commission (2014). Living well, within the limits of our planet. 7th eap — the new general union environment action programme to 2020.

European Commission (2015). Urban mobility. http://ec.europa.eu/transport/themes/urban/urban_mobility/index_en.htm.

Grangier, P., Gendreau, M., Lehuédé, F., and Rousseau, L.-M. (2016). An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. *European Journal of Operational Research*, 254(1):80–91.

Gu, L. (2018). Delivery robots hit the road in beijing. http://www.ecns.cn/news/sci-tech/2018-06-20/detail-ifyvmiee7350792.shtml. Accessed June 20, 2018.

Han, A. F.-W. and Chu, Y.-C. (2016). A multi-start heuristic approach for the split-delivery vehicle routing problem with minimum delivery amounts. *Transportation Research Part E: Logistics and Transportation Review*, 88:11–31.

Hemmelmayr, V. C., Cordeau, J.-F., and Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & operations research*, 39(12):3215–3228.

Jemai, J. and Sarkar, B. (2019). Optimum design of a transportation scheme for healthcare supply chain management: The effect of energy consumption. *Energies*, 12(14):2789.

Karak, A. and Abdelghany, K. (2019). The hybrid vehicle-drone routing problem for pick-up and delivery services. *Transportation Research Part C: Emerging Technologies*, 102:427–449.

LI, D. (2017). Jd.com launches robot delivery services in chinese universities. `https://www.chinamoneynetwork.com/2017/06/19/jd-com-launches-robot-delivery-services-in-chinese-universities`. Accessed June 19, 2017.

Li, F., Golden, B., and Wasil, E. (2007). The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Computers & Operations Research*, 34(10):2918–2930.

Li, H., Zhang, L., Lv, T., and Chang, X. (2016). The two-echelon time-constrained vehicle routing problem in linehaul-delivery systems. *Transportation Research Part B: Methodological*, 94:169–188.

Lithia, F. (2017). Ups tests residential delivery via drone launched from atop package car. `https://pressroom.ups.com/pressroom/ContentDetailsViewer.page?ConceptType=PressReleases&id=1487687844847-162`. Accessed February 21, 2017.

Liu, T., Luo, Z., Qin, H., and Lim, A. (2018). A branch-and-cut algorithm for the two-echelon capacitated vehicle routing problem with grouping constraints. *European Journal of Operational Research*, 266(2):487–497.

Luo, Z., Liu, Z., and Shi, J. (2017). A two-echelon cooperated routing problem for a ground vehicle and its carried unmanned aerial vehicle. *Sensors*, 17(5):1144.

Martí, R., Aceves, R., León, M. T., Moreno-Vega, J. M., and Duarte, A. (2019). Intelligent multi-start methods. In *Handbook of Metaheuristics*, pages 221–243. Springer.

Murray, C. and Raj, R. (2019). The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Available at SSRN 3338436*.

Murray, C. C. and Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109.

Nagy, G., Wassan, N. A., Speranza, M. G., and Archetti, C. (2013). The vehicle routing problem with divisible deliveries and pickups. *Transportation Science*, 49(2):271–294.

Nguyen, V.-P., Prins, C., and Prodhon, C. (2012). A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem. *Engineering Applications of Artificial Intelligence*, 25(1):56–71.

Otto, A., Agatz, N., Campbell, J., Golden, B., and Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. *Networks*, 72(4):411–458.

Paolo, T. and Vigo, D. (2014). *Vehicle routing: problems, methods and applications*. Society for Industrial and Applied Mathematics.

Parragh, S. N. and Cordeau, J.-F. (2017). Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. *Computers & Operations Research*, 83:28–44.

Poikonen, S. and Golden, B. (2020). Multi-visit drone routing problem. *Computers & Operations Research*, 113:104802.

Poikonen, S., Golden, B., and Wasil, E. A. (2019). A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing*, 31(2):335–346.

Poikonen, S., Wang, X., and Golden, B. (2017). The vehicle routing problem with drones: Extended models and connections. *Networks*, 70(1):34–43.

Polat, O. (2017). A parallel variable neighborhood search for the vehicle routing problem with divisible deliveries and pickups. *Computers & Operations Research*, 85:71–86.

Pugliese, L. D. P. and Guerriero, F. (2017). Last-mile deliveries by using drones and classical vehicles. In *International Conference on Optimization and Decision Science*, pages 557–565. Springer.

Repoussis, P. P., Tarantilis, C. D., and Ioannou, G. (2007). The open vehicle routing problem with time windows. *Journal of the Operational Research Society*, 58(3):355–367.

Rothenbächer, A.-K., Drexl, M., and Irnich, S. (2018). Branch-and-price-and-cut for the truck-and-trailer routing problem with time windows. *Transportation Science*, 52(5):1174–1190.

Sacramento, D., Pisinger, D., and Ropke, S. (2019). An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C: Emerging Technologies*, 102:289–315.

Sarkar, B., Ganguly, B., Sarkar, M., and Pareek, S. (2016). Effect of variable transportation and carbon emission in a three-echelon supply chain model. *Transportation Research Part E: Logistics and Transportation Review*, 91:112–128.

Schermer, D., Moeini, M., and Wendt, O. (2019a). A hybrid vns/tabu search algorithm for solving the vehicle routing problem with drones and en route operations. *Computers & Operations Research*, 109:134–158.

Schermer, D., Moeini, M., and Wendt, O. (2019b). A matheuristic for the vehicle routing problem with drones and its variants. *Transportation Research Part C: Emerging Technologies*, 106:166–204.

Scherr, Y. O., Saavedra, B. A. N., Hewitt, M., and Mattfeld, D. C. (2019). Service network design with mixed autonomous fleets. *Transportation Research Part E: Logistics and Transportation Review*, 124:40–55.

Tang, C. S. and Veelenturf, L. P. (2019). The strategic role of logistics in the industry 4.0 era. *Transportation Research Part E: Logistics and Transportation Review*, 129:1–11.

Van Woensel, T. (2018). Instances used. `https://tomvanwoensel.com/instances-used/`. Accessed June 03, 2018.

Wang, X., Poikonen, S., and Golden, B. (2017). The vehicle routing problem with drones: several worst-case results. *Optimization Letters*, 11(4):679–697.

Wang, Y., Assogba, K., Liu, Y., Ma, X., Xu, M., and Wang, Y. (2018). Two-echelon location-routing optimization with time windows based on customer clustering. *Expert Systems with Applications*, 104:244–260.

Wang, Z. and Sheu, J.-B. (2019). Vehicle routing problem with drones. *Transportation Research Part B: Methodological*, 122:350–365.

Yu, K., Budhiraja, A. K., Buebel, S., and Tokekar, P. (2018). Algorithms and experiments on routing of unmanned aerial vehicles with mobile recharging stations. *Journal of Field Robotics*, pages 1–15.

Appendix A. Column generation procedure

We design a basic column generation procedure, based on the book of column generation (**?**), to solve the original MIP model. The objective function in our paper first minimizes the number of LVs used, then keep the number of used LVs unchanged, to minimize the total travel cost. Since the CPLEX can well solve the primary objective, we only use the column generation to calculate the secondary objective lower bound. In the column generation procedure, we first keep the fixed number of LVs (primary objective got from CPLEX) unchanged and then adopt the column generation to get a lower bound of the secondary objective.

Appendix A.1 and Appendix A.2 describe the master problem and pricing sub-problem for the column generation procedure of the original MIP model. A column generation lower bound computational study for small-sized instances is presented in Appendix A.3.

Appendix A.1. Master problem

The master problem is simply stated as a set partitioning problem.

Let R be the set of all feasible tour-trees. We define a tour-tree is feasible if the capacity, time window, travel distance, and synchronization constraints are respected. For each tour-tree $r \in R$, we define c_r as the total cost for tour-tree r. Furthermore, let α_{rz} be a binary coefficient equal to 1 if the customer z is visited by the tour-tree r (0 otherwise). Let y_r denote a binary variable that takes the value 1 if and only if the tour-tree $r \in R$ is included in the solution (0 otherwise). Let K_{fix} be the fixed number of LV we used.

Based on these definitions, the LV-SAV problem formulation can be written as the following set partitioning problem:

Objective

$$\min(\sum_{r\in R} c_r y_r) \tag{A.1}$$

Subject to

$$\sum_{r\in R} \alpha_{rz} y_r = 1, \forall z \in Z \tag{A.2}$$

$$y_r \in \{0, 1\}, \forall r \in R \tag{A.3}$$

$$\sum_{r\in R} y_r = K_{fix} \tag{A.4}$$

The objective function (A.1) minimizes the total cost of the used tour-trees. Constraints (A.2) ensure each customer is served by one tour-tree. Constraints (A.3) are the domain constraint for the decision variables. Note that for implementation issues, constraints (A.3) can be replaced by $0 \leq y_r \leq 1$. Constraints (A.4) ensure the number of the vehicle used is fixed.

Appendix A.2. Pricing sub-problem

We use MIP model to solve the pricing sub-problem.

Let γ_z, $z \in Z$ be the dual variable associated with constraints (A.2). And β be the dual variable of constraint (A.4). The reduced cost $\overline{c}_r = c_r - \sum_{r\in R} \alpha_{rz}\gamma_z - \beta$. Explanation of other symbol definitions and inequalities are the same as the explanation in our LV-SAV model expect without symbol k. Which means we consider one LV, not multi-LVs. For the sake of simplicity, we will not explain more.

The pricing sub-problem, aiming to generate columns, with the most negative reduced cost, is formulated as follows:

Objective

$$\min\Big(\sum_{(i,j)\in A_1} c_{i,j} * x_{i,j} + \sum_{(i,j)\in A_2} (c_{i,j} - \gamma_i) y^l_{i,j} - \beta\Big) \tag{A.5}$$

Subject to

$$\sum_{(i,j)\in A_1} x_{i,j} \le 1, \qquad \forall j \in V_r^{'} \tag{A.6}$$

$$\sum_{i\in V_p} x_{i,0'} = \sum_{j\in V_d} x_{0,j} = 1 \tag{A.7}$$

$$\sum_{(i,j)\in A_1} x_{i,j} - \sum_{(j,i)\in A_1} x_{j,i} = 0, \qquad \forall j \in V_r \tag{A.8}$$

$$\sum_{l\in F_D} \sum_{i\in V_{dc}} y^l_{i,j} <= 1, \qquad \forall j \in V_c \tag{A.9}$$

$$\sum_{i\in V_{dc}} y^l_{i,j} - \sum_{i\in V_{pc}} y^l_{j,i} = 0, \qquad \forall j \in V_c, l \in F_D \tag{A.10}$$

$$\sum_{i\in V_p} Q_{i,0'} = \sum_{j\in V_d} Q_{0,j}, \tag{A.11}$$

$$\sum_{j\in V_d} Q_{0,j} = \sum_{l\in F_D} \sum_{i\in V_c} \sum_{j\in V_d} y^l_{j,i}, \tag{A.12}$$

$$\sum_{i\in V_r^0} Q_{i,j} - \sum_{i\in V_r} Q_{j,i} = \sum_{l\in F_D} \sum_{i\in V_c} y^l_{j,i}, \qquad \forall j \in V_d \tag{A.13}$$

$$\sum_{i\in V_r} Q_{i,j} - \sum_{i\in V_r^{'}} Q_{j,i} = -\sum_{l\in F_D} \sum_{i\in V_c} y^l_{i,j}, \qquad \forall j \in V_p \tag{A.14}$$

$$0 \le Q_{i,j} \le x_{i,j} * L, \qquad \forall (i,j) \in A_1 \tag{A.15}$$

$$W_i + t_{i,j} - W_j \le M(1 - x_{i,j}), \qquad \forall (i,j) \in A_1 \tag{A.16}$$

$$W_i + t_{i,j} - W_j \le M(1 - y^l_{i,j}), \qquad \forall i \in V_d, j \in V_c, l \in F_D \tag{A.17}$$

$$W_i + t_{i,j} + s_i - W_j \le M(1 - y^l_{i,j}), \qquad \forall i \in V_c, j \in V_{pc}, l \in F_D \tag{A.18}$$

$$a_i \le W_i, \qquad \forall i \in V_c^0 \tag{A.19}$$

$$W_i \le b_i, \qquad \forall i \in V_c^{'} \tag{A.20}$$

$$\sum_{i\in V_c} d_i \sum_{j\in V_{pc}} y^l_{i,j} \le C, \qquad \forall l \in F_D \tag{A.21}$$

$$\sum_{(i,j)\in A_2} y^l_{i,j} t_{i,j} \le T, \qquad \forall l \in F_D \tag{A.22}$$

$$x_{i,j} \in \{0,1\}, \qquad \forall (i,j) \in A_1 \tag{A.23}$$

$$y^l_{i,j} \in \{0,1\}, \qquad \forall (i,j) \in A_2, l \in F_D \tag{A.24}$$

Appendix A.3. Computational results of column generation lower bound

According to (Desrosiers and Lübbecke 2005), the lower bound can be calculate as $LB = z + K * c$, Where z denotes the optimal objective function value to the restricted master problem. c is the reduce cost got from the subproblem, and K is an upper bound of the number of LV we used. The runtime of the column generation procedure also limited to 5 hours, and the best of the valid lower bounds, each iteration generated, are chosen as the column generation lower bound. Note that the column generation procedure is code in DOcplex by calling CPLEX 12.8.

The column generation computational results for 15 and 30 customer instances are in Table A.16. In which column 1 indicates the test instances, column 2 and column 3 show the column generation lower bound and the solving time for the instance.

Table A.16: Column generation lower bound for 15 and 30 customer instances

	3-15		4-15		5-15		3-30		4-30		5-30	
	CLB	CT(s)	CLB	CT(s)	CLB	CT(s)	CLB	CT(s)	CLB	CT(s)	CLB	CT(s)
CA1	375.6	49.6	411.1	284.6	403.7	522.5	752.7	3375.3	693.7	8332.5	602.2	8484.6
CA2	378.5	80.3	403.2	115.3	418.4	698.5	653.1	13591.0	676.8	6965.7	624.8	18000.0
CA3	387.5	108.5	443.7	113.0	354.4	251.6	680.3	7756.6	639.5	18000.0	602.2	8488.1
CA4	400.9	71.4	397.8	2578.1	343.5	440.8	578.4	2837.0	676.3	18000.0	-1367.4	18000.0
CA5	382.8	52.8	308.7	102.1	386.7	963.2	650.1	9257.5	585.1	9598.6	249.6	18000.0
CB1	391.8	91.2	414.5	505.1	530.0	8060.2	758.2	16025.0	673.1	18000.0	580.1	18000.0
CB2	410.1	45.5	446.9	361.9	406.9	1224.6	626.5	4762.1	642.9	18000.0	588.1	18000.0
CB3	446.9	76.6	438.5	527.3	364.6	311.0	679.9	4886.4	623.1	18000.0	301.2	18000.0
CB4	377.6	59.8	393.6	515.0	334.2	18000.0	609.2	6910.0	654.1	18000.0	-1923.9	18000.0
CB5	408.1	89.6	361.6	127.6	393.7	941.4	621.5	18000.0	601.9	18000.0	315.8	18000.0
CC1	358.8	210.9	415.5	18000.0	332.7	18000.0	389.0	18000.0	503.3	18000.0	316.8	18000.0
CC2	344.8	1198.2	391.5	10235.8	350.7	18000.0	290.0	18000.0	377.4	18000.0	-3355.5	18000.0
CC3	380.9	976.4	419.8	18000.0	337.9	18000.0	450.8	18000.0	-2773.7	18000.0	292.3	18000.0
CC4	373.3	547.3	364.4	18000.0	307.9	18000.0	442.0	18000.0	453.8	18000.0	-3453.5	18000.0
CC5	286.5	5557.1	301.4	7705.9	374.4	18000.0	432.7	18000.0	272.9	18000.0	-983.8	18000.0
CD1	370.8	71.3	405.3	395.8	385.4	792.1	619.7	13819.9	671.2	4944.2	624.5	18000.0
CD2	348.8	95.5	396.9	258.3	384.2	828.9	624.8	12195.6	643.9	9956.6	612.7	18000.0
CD3	382.1	116.8	445.0	403.4	348.3	320.5	645.7	4853.2	606.5	18000.0	588.3	9042.9
CD4	381.6	114.0	355.4	780.9	337.3	585.1	578.4	1612.6	661.7	18000.0	630.1	18000.0
CD5	362.5	60.0	301.4	7790.1	393.2	520.3	641.9	5025.1	630.5	7475.1	441.5	18000.0