Multi-depot multi-trip vehicle routing problem with time windows and release dates

Abstract: This study investigates a multi-depot multi-trip vehicle routing problem with time windows and release dates, which is a practical problem in the last mile distribution operations. This problem aims to design a set of trips for the fleet of vehicles supplied by different depots for minimizing total traveling time. It addresses some realistic considerations, such as the customers' time windows and the release date of customers' packages. The problem is formulated as a mixed integer programming model. A hybrid particle swarm optimization algorithm and a hybrid genetic algorithm are developed to solve this problem. Extensive numerical experiments are conducted to validate the effectiveness of the proposed model and the efficiency of the proposed solution methods. The experimental results show that our proposed algorithms can obtain near-optimal solutions for small-scale problem instances, and solve some large-scale instances with up to 200 customers, 20 depots and 40 vehicles in reasonable computation time.

Keywords: Vehicle routing; release date; multi-depot; multi-trip; time window.

1. Introduction

Due to the rapid development of electronic commerce (e-commerce), increasing customers prefer to shop online rather than onsite. In China, the number of packages delivered has reached 40 billion (China State Post Bureau, 2018). This remarkable increase raises serious requirements for the levels of logistics services. However, online shopping customers normally complain about package delays and inconsistent delivery services. Therefore, the last mile distribution is very critical for improving the online shopping experience by customers.

The last mile distribution has the characteristics of dispersive customer positions, large order volumes, small batches, and multiple repeated routes. In this case, vehicles usually deliver packages from multiple distribution centers to multiple customers every day. In addition, due to the limited capacity and the goal of minimizing fleet size, they must operate multiple trips. For the demand side, customers typically specify service time intervals, i.e., time windows. Meanwhile, the release dates of packages to be delivered are normally imposed, which follow the practices that vehicles can only deliver available packages that have been released by the distribution center. Note that the release date introduced in this paper is not the same as the release date considered in the scheduling problem. The latter is normally referred to as ready time, which is the earliest time to process work on a machine (Pinedo, 2005). Our release date is the time when the package requested by a customer in the distribution center can be delivered by a vehicle (Cattaruzza et al., 2016).

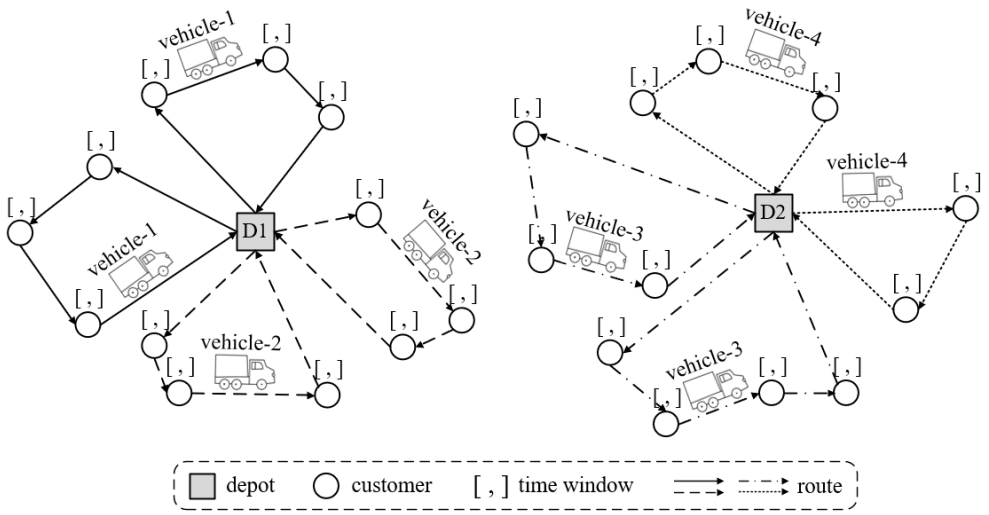


Figure 1: A case of multi-depot multi-trip VRP with time windows and release dates

In this paper, we define the above problem as a multi-depot multi-trip vehicle routing problem with time windows and release dates (Multi-D&T VRPTW-R, as shown in *Figure 1*). We first formulate it as a mixed integer programming (MIP) model. By using the model, some off-the-shelf solvers, e.g., CPLEX, can only solve small-scale instances. Thus, we develop a hybrid particle swarm optimization algorithm (HPSO) and a hybrid genetic algorithm (HGA) to solve large-scale instances effectively, which are also compared by conducting extensive numerical experiments. The results show that our algorithms can solve large-scale instances

with up to 200 customers, 20 depots and 40 vehicles effectively. They can also derive near-optimal solutions for small-scale problem instances.

The remainder of this paper is organized as follows. Section 2 reviews related works. Section 3 defines the problem and formulates it as a mathematical model. Section 4 proposes two heuristic algorithms. Section 5 provides the results of numerical experiments. Conclusions are outlined in the last section.

2. Related works

The vehicle routing problem (VRP) was first proposed by Dantzig and Ramser (1959), aiming at selecting optimal routes for vehicles that serve a set of customers. The classic VRP has several assumptions as follows: there is a single depot; each vehicle must depart from and return to the depot, and each customer can be served exactly once by a vehicle. General and state-of-the-art surveys about the VRP can be referred to Laporte (2009), Toth and Vigo (2014). A well-known variant of VRP is to consider the requirements of the vehicles' arrival time at customers and time windows restrictions. This variant is defined as the vehicle routing problem with time windows (VRPTW). Many scholars have conducted advanced studies in the fields of the VRPTW (Desrochers et al. 1992, Bent et al. 2004, Alvarez et al. 2017). A popular variant is to consider both the pickup and the delivery in vehicle routing, Nagy et al. (2015) studied the VRP that allows divisible deliveries and pickups, and compare the savings with respect to the case where the cargos must be served simultaneously.

Multiple depots VRP (Multi-D VRP) is an extension of the classical VRP. In the problem, vehicles serve customers based on multiple depots and the tours of vehicles must begin and end at the same depot. Salhi et al. (2013) studied the multi-depot vehicle routing problem in consideration of heterogeneous vehicle fleet, they developed a variable neighborhood search approach and the experiments indicated that the approach is highly competitive. For the research of combining multiple depots and VRPTW, it can be referred to Desaulniers et al. (1998), Dondo and Cerdá (2009), Bettinelli et al. (2011), Bae and Moon (2016). Desaulniers et al. (1998) proposed a multi-depot VRPTW (Multi-D VRPTW) where a fleet of vehicles are

provided by different depots and the tasks performed by a group of vehicles are limited to start within a time interval. This problem is formulated as an integer nonlinear multi-commodity network flow model and is solved by using a column generation method embedded in a branch-and-bound framework that basically forms a branch-and-price scheme. Dondo and Cerdá (2009) developed a local search improvement algorithm to solve the Multi-D VRPTW, which aims to explore large neighborhoods of the current solutions to obtain a set of minimum-cost routes. Their method can solve large-scale benchmark problems with up to 200 customers. Bettinelli et al. (2011) presented a branch-and-cut-and-price algorithm to discover the exact solution of a variation of the VRPTW, where vehicles are limited by different capacities and depart from different depots. Bae and Moon (2016) applied the Multi-D VRPTW to a practical, challenging logistics and supply chain management problem, i.e., using vehicles for delivery and installation of electronic equipment.

The multi-trip VRP (Multi-T VRP) is a variant of the classical VRP, which considers that each vehicle can travel for multiple trips. Wassan et al. (2017) studied the multi-trip vehicle routing problem with backhauls and developed a two-level variable neighborhood search algorithm. Research on integrating multi-trip and VRPTW can be found in Nguyen et al. (2013), Yan et al. (2015) and Hernandez et al. (2016). Nguyen et al. (2013) proposed a tabu search metaheuristic for the time-dependent multi-zone multi-trip VRPTW, which is used to improve route selection and vehicle route allocation. Their experimental results indicated that their method can yield higher quality solutions in terms of both the required number of vehicles and traveling cost. Yan et al. (2015) investigated a multi-trip split delivery VRPTW, allowing multiple uses of vehicles. They proposed a two-step solution algorithm to solve this NP-hard problem. Their results illustrated that these VRPs can be solved effectively and efficiently. Hernandez et al. (2016) addressed a multi-trip VRPTW (Multi-T VRPTW) by considering the case that vehicles must visit all customers, and the duration is unlimited. Their problem was solved by two branch-and-price frameworks based on two-set covering formulations. They evaluated and compared these two methods by conducting numerical experiments for small-scale (25 customers) benchmark instances. Tang et al. (2015) proposed an exact algorithm

based on the trip-chain-oriented set-partitioning model to solve the multi-trip vehicle routing and scheduling problem.

Release dates have been considered by a few VRP studies. Archetti et al. (2018) studied the traveling salesman problem (TSP) with release dates, introduced some properties, and proposed an iteration based local search approach. Archetti et al. (2015) and Reyes et al. (2018) introduced release dates in the VRP where vehicles belong to a single depot and their capacity is not restricted. Shelbourne et al. (2017) introduced a novel extension of vehicle routing and scheduling problem, integrated aspects of machine scheduling into vehicle routing, and proposed a path-relinking algorithm with neighborhood search. Inspired by the city logistics field, Cattaruzza et al. (2016) proposed a Multi-T VRPTW considering release dates (Multi-T VRPTW-R) in which time windows and release dates are associated with customers. They developed a genetic algorithm to solve this problem and demonstrated the effectiveness of the method. However, they did not provide a mathematical model for their problem.

To the best of our knowledge, there is no literature devoted to a Multi-D&T VRPTW-R. The contributions of this paper are as follows. First, we propose the Multi-D&T VRPTW-R, which is based on the practical operations of online shopping package delivery. Second, we formulate this problem as a MIP model with the objective of minimizing the total traveling and service time of all vehicles. Third, an HPSO and an HGA are developed to solve the problem.

3. Problem description and formulation

In this section, we formally define the Multi-D&T VRPTW-R and then formulate a MIP for the problem.

3.1 Problem description

The Multi-D&T VRPTW-R can be defined on a directed network $G = (V, E)$, where V is the set of vertices that is partitioned into a set of depots D and a set of customers N, and $E = \{(i, j)|i, j \in V\}$ is an edge set. Let a set of vehicles as K of capacity Q and a set of trips as W. Each depot has its own fleet of vehicles. Each customer $i \in N$ has a demand q_i that should be delivered during the time window $[e_i, l_i]$. Each vehicle serving customer $i \in N$ and

travelling from $i \in N$ to $j \in V$ incurs a spent time $t_i^S + t_{i,j}^N$ (if $j \in N$) or $t_i^S + t_{k,i}$ (if $j \in D$), where t_i^S is customer service time, $t_{i,j}^N$ is the travel time between customer $i \in N$ and customer $j \in N$, and $t_{k,i}$ is the travel time between vehicle k's depot and customer $i \in N$. Let d_k represent the depot that vehicle k initially located at. The time when a vehicle departs from a depot for delivering packages must be later than the release dates t_i^R requested by all customers included in the trip of the vehicle, prior to which a laden vehicle needs to be loaded at the depot for a time t_k^D that denotes service time at vehicle k's depot. We introduce a parameter T^H representing a time horizon and thus set a time window $[0, T^H]$ for each depot, which means that each trip must start no earlier than 0 and end no later than T^H. The above important time points of Multi-D&T VRPTW-R can be seen in *Figure 2*, the focused depot is assumed as d.

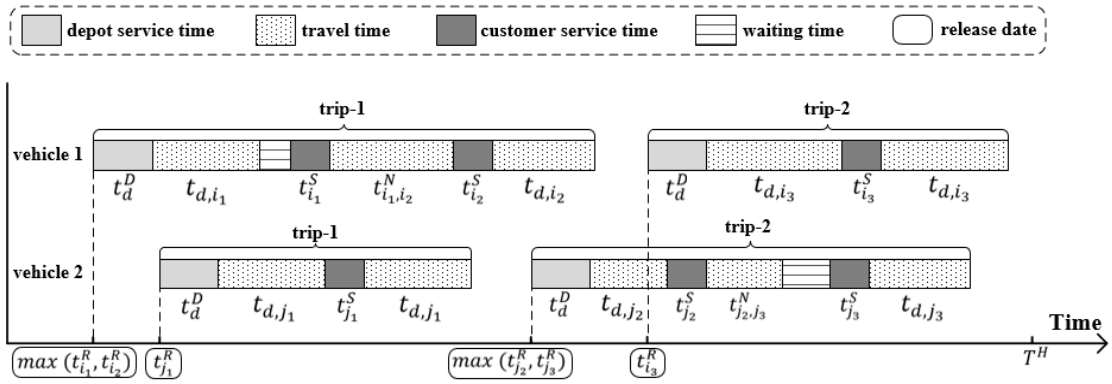


Figure 2: Important time points for the Multi-D&T VRPTW-R

Before formulating the mode of the Multi-D&T VRPTW-R, some assumptions are listed as follows:

- All fleet of vehicles are homogeneous and are available at depots.
- Each trip must start and end in the same depot.
- No overlap of time between each trip of the same vehicle.
- Each customer is served exactly once by one vehicle.
- The total demand served on the trip of a vehicle cannot exceed the vehicle's capacity.

3.2 Model formulation

In this section, we formulate the Multi-D&T VRPTW-R as a MIP model.

3.2.1 Notations

Indices and sets:

i, j, l	index of a customer.
N	set of customers.
k	index of a vehicle.
K	set of vehicles.
w	index of a trip.
W	set of trips for any vehicle.

Note: For any vehicle, the number of the real performed trip(s) gradually increases from 1 with a step size of 1. For instance, for two vehicles k and k' with two and three trips, respectively, the real performed trip sets are {1,2} and {1,2,3} respectively.

Parameters:

e_i	the earliest service time at customer i.
l_i	the latest service time at customer i.
q_i	the demand for customer i.
t_i^R	release date of packages requested by customer i.
t_i^S	service time of customer i.
$t_{i,j}^N$	travel time between customer i and j.
t_k^D	service time of vehicle k's depot.
$t_{k,i}$	travel time between vehicle k's depot and customer i.
T^H	time horizon.
Q	the capacity of each vehicle.
M	a sufficiently large positive number.

Decision variables:

$\tau_{k,w}^{L}$ time point that vehicle k departs from the depot in trip w.

$\tau_{k,w}^{B}$ time point that vehicle k returns to the depot in trip w.

$\tau_{k,i,w}$ time point that vehicle k arrives at customer i in trip w.

$\delta_{k,w}^{L}$ cargo load of vehicle k in trip w when departing from the depot.

$\delta_{k,i,w}$ cargo load of vehicle k in trip w after vehicle k visiting customer i.

$\gamma_{k,w}$ binary equals one if vehicle k operates trip w; otherwise, equals zero.

$\xi_{i,j,k,w}$ binary equals one if vehicle k visits customer j immediately after visiting customer i in trip w; otherwise, equals zero.

$\lambda_{k,i,w}$ binary equals one if vehicle k serves customer i in trip w; otherwise, equals zero.

$\mu_{k,i,w}$ binary equals one if customer i is the first customer served by vehicle k in trip w; otherwise, equals zero.

$\eta_{k,i,w}$ binary equals one if customer i is the last customer served by vehicle k in trip w; otherwise, equals zero.

3.2.2 Mathematical model

$$\textit{Minimize } Z = \sum_{k\in K}\sum_{w\in W}\sum_{i\in N}\sum_{j\in N} t_{i,j}^{N}\xi_{i,j,k,w} + \sum_{k\in K}\sum_{w\in W}\sum_{i\in N} t_{k,i}\mu_{k,i,w} + \sum_{k\in K}\sum_{w\in W}\sum_{i\in N} t_{k,i}\eta_{k,i,w} \quad (1)$$

Subject to:

$$\sum_{k\in K}\sum_{w\in W}\lambda_{k,i,w} = 1 \qquad \forall\, i \in N \quad (2)$$

$$\mu_{k,l,w} + \sum_{i\in N}\xi_{i,l,k,w} = \eta_{k,l,w} + \sum_{j\in N}\xi_{l,j,k,w} = \lambda_{k,l,w} \qquad \forall\, l \in N, k \in K, w \in W \quad (3)$$

$$\sum_{i\in N}\mu_{k,i,w} = \sum_{i\in N}\eta_{k,i,w} = \gamma_{k,w} \qquad \forall\, k \in K, w \in W \quad (4)$$

$$\gamma_{k,w+1} \le \gamma_{k,w} \qquad \forall\, k \in K, w \in W/|W| \quad (5)$$

$$\sum_{i\in N}\lambda_{k,i,w} \le \gamma_{k,w}|N| \qquad \forall\, k \in K, w \in W \quad (6)$$

$$\delta_{k,w}^{L} = \sum_{i\in N}\lambda_{k,i,w}q_i \qquad \forall\, k \in K, w \in W \quad (7)$$

$$\delta_{k,w}^{L} \le \gamma_{k,w}Q \qquad \forall\, k \in K, w \in W \quad (8)$$

$$\tau_{k,i,w} \le \lambda_{k,i,w}l_i \qquad \forall\, i \in N, k \in K, w \in W \quad (9)$$

$$\tau_{k,w}^{L} \ge \lambda_{k,i,w}(t_i^R + t_k^D) - M(1-\gamma_{k,w}) \qquad \forall\, i \in N, k \in K, w \in W \quad (10)$$

$$\tau_{k,w+1}^{L} \ge \tau_{k,w}^{B} + t_k^D - M(1-\gamma_{k,w+1}) \qquad \forall\, i \in N, k \in K, w \in W \quad (11)$$

$$\tau_{k,i,w} \geq \tau_{k,w}^{L} + t_{k,i} - M(1 - \mu_{k,i,w}) \qquad \forall\, i \in N, k \in K, w \in W \qquad (12)$$

$$\tau_{k,w}^{B} \geq max\{\tau_{k,i,w}, e_i\} + t_i^S + t_{k,i} - M(1 - \eta_{k,i,w}) \qquad \forall\, i \in N, k \in K, w \in W \qquad (13)$$

$$\tau_{k,j,w} \geq max\{\tau_{k,i,w}, e_i\} + t_i^S + t_{i,j}^N - M(1 - \xi_{i,j,k,w}) \qquad \forall\, i,j \in N, k \in K, w \in W \qquad (14)$$

$$\tau_{k,w}^{B} \leq T^H \qquad \forall\, k \in K, w \in W \qquad (15)$$

$$\gamma_{k,w} \in \{0,1\} \qquad \forall\, k \in K, w \in W \qquad (16)$$

$$\xi_{i,j,k,w} \in \{0,1\} \qquad \forall\, i,j \in N, k \in K, w \in W \qquad (17)$$

$$\lambda_{k,i,w} \in \{0,1\} \qquad \forall\, i \in N, k \in K, w \in W \qquad (18)$$

$$\mu_{k,i,w} \in \{0,1\} \qquad \forall\, i \in N, k \in K, w \in W \qquad (19)$$

$$\eta_{k,i,w} \in \{0,1\} \qquad \forall\, i \in N, k \in K, w \in W \qquad (20)$$

The objective function (1) minimizes the total traveling time of all the vehicles. Constraints (2) impose that each customer is served exactly once. Constraints (3) show that, for a customer who is served in a trip of a vehicle, if he/she is the first customer in the served customer sequence of this trip, then no customer can be served before him/her in this trip, otherwise, there must exist a customer who is served exactly before him/her. Similarly, if he/she is the last customer in the served customer sequence of this trip, then no customer can be served after him/her in this trip, otherwise, there must be a customer who is served exactly after him/her. Constraints (4) ensure that there must exist only one first customer point and one last customer point in a trip that is performed, on the contrary, if the trip is not performed, then no first customer point exists in this trip, nor the last customer point. Constraints (5) guarantee that trip $w + 1$ of vehicle k can be operated only if vehicle k has operated trip w. Constraints (6) guarantee that a vehicle can serve a customer only if the customer is included in this vehicle's trip. Constraints (7) ensure that the total loads of a vehicle equal the sum of customer demands. Constraints (8) ensure that the total loads of a vehicle cannot exceed its vehicle capacity. Constraints (9) are the service time window constraints. Constraints (10) address the time relationship between the departure time of a vehicle and release dates as well as depot service time in a trip. Constraints (11) state the time relationship between consecutive trips of the same vehicle. Constraints (12)–(13) are travel time conservation constraints. Constraints (14) are service time conservation constraints. Constraints (15) ensure that the time that a vehicle is

returned to the depot is no later than the time horizon T^H. Constraints (16)–(20) define the domains of decision variables.

It should be noted that constraints (11), (13), (14) are presented in a nonlinear form with *max* functions, there is no necessity to linearize them because the current version of the model implementation tools applied in this study (introduced in Section 5) already support the *max/min* functions.

4. Algorithmic strategies

The model developed in Section 3 can be solved by some off-the-shelf solvers (e.g., CPLEX) for small-scale instances. To deal with large-scale instances, we develop an HPSO algorithm and an HGA algorithm to solve our proposed model and compare their performance through extensive experiments. The classical particle swarm optimization (PSO) algorithm initializes a set of positions of particles (i.e., the swarm), and obtains the optimal position by iteratively updating the positions towards the incumbent best one. In our strategies, HPSO combines the local search-variable neighborhood descent (LS-VND) and the solution translation mechanism to expand search neighborhoods as well as accelerate the algorithm. The classical genetic algorithm (GA) initializes a set of chromosomes (i.e., the population) and the best individual information is found through evolution and iteration mechanisms. In our strategies, a reorder routine is adapted to accelerate our HGA.

In Section 4.1 and 4.2, we separately describe the frameworks for the proposed HPSO and HGA. There are several same procedures in HPSO and HGA, such as LS-VND, labeling procedure. The details of these procedures are all introduced as follows.

4.1 HPSO Algorithm

In HPSO, particles in swarm represent solutions. Each particle is identified by a customer id sequence (sorted by increasing order) and a corresponding *random position* sequence (see *Figure 3(a)*). Then, a labeling procedure is executed to evaluate the fitness value of each particle,

Algorithm 1: HPSO framework

```
1:  initialize swarm                                   //section 4.1.2
2:  for each particle
3:      fitness evaluation (Labeling procedure)        //section 4.1.3
4:      improve solutions (LS-VND)                     //section 4.1.4
5:      solution translation                           //section 4.1.5
6:      update best particle cost and best global cost
7:  end for
8:  while termination criteria are not met
9:      update particle random position                //section 4.1.6
10:     go to Line 2
11: end while
```

as well as to obtain the trips' assignments to vehicles. Next, the LS-VND is executed for each particle to find a better customer sequence by adjusting the order of customers in the sequence. If the customer sequence in a particle is modified following the sequence obtained by LS-VND, a translation between different representations of this particle will be performed, where we define it as the solution translation mechanism. When the termination criteria are met, the iteration is stopped and the incumbent best solution is output. The outline of the method (i.e., HPSO) is given in Algorithm 1.

4.1.1 Solution representation of HPSO

In this study, each particle is structured as a two-dimensional array in length of $|N|$, denoted as particle array Ap. The first array $Ap(1,:)$ indicates the customer id, while the second array $Ap(2,:)$ indicates the *random position* of the correspond customers in $Ap(1,:)$ and falls in the ranged of $[0,|D|)$. The values of $Ap(2,j)$ indicates the *random position* associated to the j^{th} customer in $Ap(1,:)$. *Figure 3(a)* shows a possible particle example of 10 customers and 2 depots, *Figure 3(b)* shows the two customer sequence of the two depots, the transformation process from a particle to customer sequence of depots is explained next.

4.1.2 Initialization of HPSO

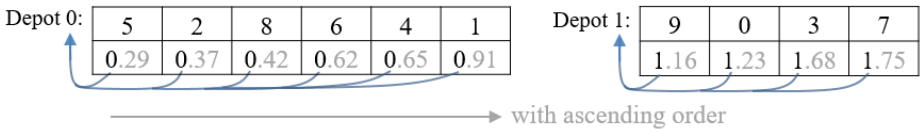
In the proposed HPSO, the initial positions of particles are randomly generated during the initialization procedure. For instance, there are $|N|$ customers and $|D|$ depots. Each customer is associated with a randomly generated *random position* between $[0,|D|)$. The position of this particle is characterized by its *random position* sequence. When a particle is generated (e.g.,

Figure 3(a)), it determines the depot that serves each customer according to the customer's *random position*. Denote *d* as the integer value of customer j^{th} *random position*, therefore this customer is assigned to be served by depot *d*. Therefore, the customer sequence of each depot can be obtained by decoding the ascending order of the *random position* (e.g., Figure 3(b)).

Figure 3 depicts an example of the initialization procedure of HPSO, whereas $|N|$ equals 10, and $|D|$ equals 2. Particle with the *random position* for customers is shown in Figure 3(a). Customers in $\{1, 2, 4, 5, 6, 8\}$ are assigned to depot $\{0\}$ as the integer value of their *random position* is 0. Similarly, customers in $\{0, 3, 7, 9\}$ are assigned to be served by depot 1. Therefore, the customer sequences of the two depots are depicted as $\{5, 2, 8, 6, 4, 1\}$ and $\{9, 0, 3, 7\}$, respectively (see Figure 3(b)).

customer id	0	1	2	3	4	5	6	7	8	9
random position	1.23	0.91	0.37	1.68	0.65	0.29	0.62	1.75	0.42	1.16

(a) HPSO particle



(b) Customer sequence of each depot

Figure 3: Particle initialization and customer sequence of depots

4.1.3 Labeling procedure

Referring to the third line in Algorithm 1, the labeling procedure is to obtain traveling routes and assign trips to vehicles.

The labeling procedure is an evaluation procedure proposed by Diego (2015). In this study, the development of labeling procedure is based on Cattaruzza et al. (2016), it transfers the customer sequence of a particle into a Multi-D&T VRPTW-R's solution and it evaluates the fitness value of this solution.

For a customer sequence of a depot, the procedure starts with the first customer in the sequence and generates labels for the customer, then a part of the labels with relatively poor quality in this customer is eliminated through elimination mechanism. Similar processing is then performed on subsequent customers in the sequence in turn (i.e., generates labels and then

eliminates a portion of them). Each label contains $(nv_d + 4)$ elements, where nv_d is the number of vehicles in depot d. The first nv_d elements indicate the available time of the nv_d vehicles to start the next trip, and is sorted in decreasing order in a label, the $(nv_d + 1)^{th}$ element indicates the predecessor node (depot or customer) , the $(nv_d + 2)^{th}$ element indicates the cost of the label, the $(nv_d + 3)^{th}$ element indicates whether it is eliminated, and the $(nv_d + 4)^{th}$ element indicates its feasibility. Note that the assignments through labelling procedure based on a customer sequence may be infeasible. This is due to the occurrence of time window and load violations when traveling routes are generated during labelling procedure. Time window and load violations are allowed and are penalized in the fitness value of solution.

Fitness evaluation of a particle in the proposed HPSO is to calculate the penalized cost for each depot separately at first, and then sum them up. Regarding the penalty criterion of time windows, we adopt the method proposed by Nagata, Bräysy, and Dullaert (2010). The penalized cost of each depot contains three parts: the traveling time of vehicle fleet, the penalization by time window violations of the customers and the penalization by load violations of the vehicle fleet. As stated above, the depot that serves each customer is decided when a particle is generated. For each depot d, there is a corresponding customer sequence denoted by S_d, and a cost function for a sequence S_d denoted by $C(S_d)$. $DS(\rho)$ is traveling time of trip ρ, $TV(\rho)$ is the amount of time window violation in trip ρ, $LV(\rho)$ is the amount of load violation of vehicle in trip ρ, and W_d is the number of all available trips starting from depot d. Thus, the fitness $R(\Psi)$ of a particle Ψ can be obtained by (21), (22).

$$C(S_d) = \sum_{\rho=1}^{W_d} DS(\rho) + \theta \sum_{\rho=1}^{W_d} TV(\rho) + \lambda \sum_{\rho=1}^{W_d} LV(\rho) \tag{21}$$

$$R(\Psi) = \sum_{d=1}^{|D|} C(S_d) \tag{22}$$

Many labels will be produced with the increasing of problem scale and it slows down the computation performance based on some preliminary tests. To circumvent this situation, a weak elimination mechanism is adapted as an acceleration method to speed up labeling procedure and the dominated labels will be removed during the labeling procedure.

Elimination mechanism: label ς^1 dominates ς^2 when conditions (23) and (24) are met.

$$C(\varsigma^1) + \theta \sum_{j=1}^{nv_d} \sigma_j(\varsigma^1, \varsigma^2) \leq \gamma C(\varsigma^2) \tag{23}$$

$$C(\varsigma^1) \leq C(\varsigma^2) \tag{24}$$

$$\sigma_j(\varsigma^1, \varsigma^2) = max\ (0, T_j(\varsigma^1) - T_j(\varsigma^2)) \tag{25}$$

where $\gamma \geq 1$.

Parameter $C(\varsigma^1)$ is the cost of the label ς^1. $T_j(\varsigma^1)$ is the j^{th} element of label ς^1, which means the available time for a vehicle to run a next trip. Parameter γ will be dynamically adjusted based on the number of labels for this customer node, and equation (26) is adapted to update γ.

$$\gamma = \begin{cases} \gamma + \frac{|\vartheta_i|}{1000\vartheta_{thr}} & if\ |\vartheta_i| > \vartheta_{thr} \\ \gamma - \frac{\vartheta_{thr}}{1000|\vartheta_i|} & if\ |\vartheta_i| < \vartheta_{thr} \end{cases} \tag{26}$$

where $|\vartheta_i|$ is the number of labels on node i produced by the label procedure, and ϑ_{thr} is a threshold parameter that indicates the number of labels. Based on some tentative experiments, it is found that the larger is γ, the solution speed is faster; while, the smaller is γ, the solution quality is better. Note that the elimination mechanism is only conducted for comparison between labels of the same customer node in the customer sequence. A special case is that there may be no label that is not dominated by the other labels for the same customer, then a preferring method is adopted to select the lowest *Cost* label among feasible ones for reservation.

At the end of labeling procedure, the lowest cost label in non-eliminated labels in the last customer node is selected, which records the associated information: the cost of the solution, the feasibility results, and the distribution plan. If there is more than one label that has the same lowest cost, the feasible label of lowest cost labels will be selected. Another possible situation is that there are more than one labels that have the same lowest cost, but they are all feasible or infeasible, in this case, we select the first generated one. An example of the labeling procedure is given in Appendix 1 attached to this paper.

Although the distribution assignment obtained through labeling procedure may be infeasible by assigning a penalized cost as the fitness to the particle (individual), it is possible to make

the superior or inferior comparison between the swarm (population) instead of simply assigning a specified value that may be very large. The latter method is hard to assess the pros and cons of particles (individuals).

4.1.4 LS-VND

Referring to the fourth line in Algorithm in Section 4.1, the purpose of LS-VND is to systematically change the set of neighborhood structures of the current solution during the search process to expand the search scope and to find a better solution.

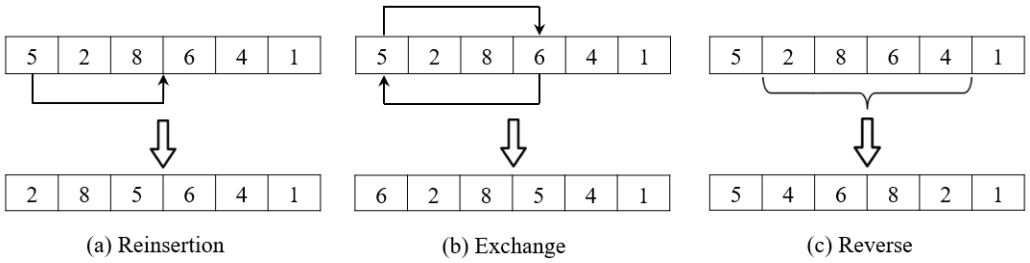


Figure 4: Three methods of local search

For each customer sequence associated with depots, we perform the LS-VND algorithm to search neighborhoods. Local search, a kind of sequence search algorithms, is used to search three basic neighborhood structures, which is described as follows:

- Reinsertion refers to relocating a customer node from one location to another location in a customer sequence, as shown in Figure 4(a), the set of the neighborhood in this structure is defined as Nh_1.
- Exchange two customer nodes from two locations in a customer sequence, as shown in Figure 4(b), the set of the neighborhood in this structure is defined as Nh_2.
- Reverse the nodes in a part of a customer sequence, as shown in Figure 4(c), the set of the neighborhood in this structure is defined as Nh_3.

Algorithm 3: LS-VND framework

1: $S_0 \leftarrow$ initial sequence; $k \leftarrow 1$
2: **while** $k \leq 3$
3: $S' \leftarrow$ find the best neighborhood of S_0 in Nh_k
4: **if** $R(S') < R(S_0)$
5: $S_0 \leftarrow S'$, $k \leftarrow 1$
6: **else**

7: $k \leftarrow k + 1$
8: end while

For each new customer sequence generated by the LS-VND, the labeling procedure is used to evaluate the cost of these sequences. Further, we update customer sequences according to their cost. From algorithm aspect, the adaption of LS-VND aims at enhancing both HPSO's intensification and diversification. The core idea of the LS-VND algorithm is to systematically change neighborhood structures of the current solution and expand the search range, and then a current best solution can be found through the local search algorithm.

4.1.5 Solution translation mechanism

Referring to the fifth line in Algorithm 1 in Section 4.1, the purpose of solution translation mechanism is to return the current best sequence and performs a translation between different representations of a particle to speed up the process. As explained in section 4.1.1, a particle is encoded as a particle array, which contains a customer id sequence and a corresponding *random position* sequence. Solution translation mechanism can be seen as a simple translation between different representations of a particle. For each depot, if a better permutation of customer sequence is obtained through LS-VND, solution translation mechanism is used to re-match the position sequence with the new permutated customer sequence. For example, Figure 3(c) gives a customer sequence $\{5, 2, 8, 6, 4, 1\}$ (denoted by CS-11) for depot $\{0\}$, the position sequence corresponds to CS-11 is $\{0.29, 0.37, 0.42, 0.62, 0.65, 0.91\}$ (denoted by PS-1). Similarly, customer sequence $\{9, 0, 3, 7\}$ (denoted by CS-21) for depot $\{1\}$ with the corresponding position sequence $\{1.16, 1.23, 1.68, 1.75\}$ (denoted by PS-2) (see Figure 5(a)). Suppose that through LS-VND, the best customer sequence of the two depots are changed as $\{2, 1, 6, 8, 5, 4\}$ (denoted by CS-12) and $\{7, 0, 9, 3\}$ (denoted by CS-22), respectively. Solution translation mechanism re-matches the original position sequence PS-1 with the new customer sequence CS-12, and PS-2 with CS-22 (see Figure 5(b)). That is to say, the position sequence before LS-VND is preserved, and the *random position* combined with customers are translated. Thus a translated particle is obtained and it changes from Figure 3(a) to Figure 5(c). In the particle

swarm iteration, the *random position* of the particles will be updated from the translated particle position of the previous generation to the new position of the new generation.

Depot 0:

CS-11	5	2	8	6	4	1
PS-1	0.29	0.37	0.42	0.62	0.65	0.91

Depot 1:

CS-21	9	0	3	7
PS-2	1.16	1.23	1.68	1.75

(a) Before LS-VND

Depot 0:

CS-12	2	1	6	8	5	4
PS-1	0.29	0.37	0.42	0.62	0.65	0.91

Depot 1:

CS-22	7	0	9	3
PS-2	1.16	1.23	1.68	1.75

(b) After LS-VND

Customer id	0	1	2	3	4	5	6	7	8	9
Position	1.23	0.37	0.29	1.75	0.91	0.65	0.42	1.16	0.62	1.68

(c) Translated particle through LS-VND

Figure 5: Solution translation mechanism in HPSO

4.1.6 Particle position update

In this section, we introduce a scheme of particle position update in HPSO. The classical PSO algorithm is suitable for solving complicated optimization problems, since it has no strict requirements on the continuity of the problem definition, and the algorithm is easy to converge.

When modifying the position of a particle during iteration, a flight direction (as well as an amplitude) in the search space is defined as velocity (Potvin, 2009). It is obtained through the randomly weighted sum of three components: the velocity of the particle in former iteration, the individual optimal of the particle with the position $pbest$ and global optimal of the swarm with the position $gbest$, the particle will update its own velocity (v) and new position (x, the same as $Ap(2, :)$ as metioned in section 4.1.1) according to the following Equation (27) and (28) (Eberhart and Kennedy,1995). Let v_{id}^{k} represent the d-dimensional component of the flight velocity vector of the k-th iteration particle i, and x_{id}^{k} is the d-dimensional component of the position vector of the k-th iteration particle i. c_1 and c_2 are two acceleration constants used to adjust the learning maximum step. r_1 and r_2 are two random numbers, ranging from zero to one. w is inertia weight, used to adjust the search space for the solution space

$$v_{id}^{k} = wv_{id}^{k-1} + c_1 r_1 \left(pbest_{id} - x_{id}^{k-1}\right) + c_2 r_2 (gbest_d - x_{id}^{k-1}) \tag{27}$$

$$x_{id}^{k} = x_{id}^{k-1} + v_{id}^{k-1} \tag{28}$$

Equation (27) represents the d-dimensional velocity update formula for particle i, and formula (28) represents the d-dimensional position update formula of particle i. The velocity of each particle will be limited to a maximum velocity V_{max}. If the update velocity of a dimension exceeds V_{max}, the velocity of this dimension is set to V_{max}.

4.2 HGA Algorithm

In this section, we introduce some specific mechanisms to implement the HGA (the other algorithm that we propose), such as the generation of offspring individuals, survivor selection of excellent individuals (also called elimination rule) in the classical genetic algorithm are considered in HGA. The classical genetic algorithm has a good global search capability and is suitable for searching complex and non-linear problems.

This section outlines our HGA for the Multi-D&T VRPTW-R. Our HGA is based on traditional GA. Moreover, we adopt some mechanisms in the genetic algorithm from Vidal et al. (2012) such as the fitness evaluation and the survivor selection, together with our proposed solution translation mechanism to achieve a significant improvement in the computation performance.

First, we initialize the population of π individuals, and each individual is a chromosome that represents the customer sequence and randomly determines which deport serves each customer. The outline of HGA is shown in Algorithm 2.

Algorithm 2: HGA framework

```
1:  initialize a population                                   // section 4.2.1
2:  while Termination criteria are not met
3:      select parent chromosomes ψ_P1 and ψ_P2
4:      generate a child ψ_C                                  // section 4.2.2
5:      fitness evaluation (Biased fitness function)          // section 4.2.3
6:      improve solutions (LS-VND)
7:      reorder routine                                       // section 4.2.4
8:      update best global cost
9:      insert ψ_C in the population
10:     if the dimension of the population reaches (π + μ)
11:         select π survivors                                // section 4.2.5
12:     endif
13: end while
```

4.2.1 Initialization of HGA

Referring to line 1 in Algorithm 2, for the population with π individuals, we randomly generate integers ranged in $[0, |D|)$ for customers in each individual to represent the mapping between customers and depots. That means, if randomly generated integer for a customer is 'd', then the customer is assigned to be served by depot d, where $d \in [0, |D|)$. Then, the customers are reordered by taking the customers of one depot together with left-shift operation.

Customer id	0	1	2	3	4	5	6	7	8	9
Integer	1	0	0	1	0	0	0	1	0	1

(a) Randomly integer for customers

1	2	4	5	6	8	0	3	7	9
0						1			

(b) Chromosome

Depot 0:	1	2	4	5	6	8	Depot 1:	0	3	7	9

(c) Customer sequence of each depot

Figure 6: Chromosome initialization in HGA

Figure 6 shows the initialization process of HGA. Figure 6(a) represents a possible randomly generated integer for customers and Figure 6(b) represents the chromosome, the corresponding customer sequence of each depot as shown in Figure 6(c). Here, $|N|$ equals 10, and $|D_k|$ equals 2. In Figure 6(b), customers $\{1, 2, 4, 5, 6, 8\}$ are assigned to depot $\{0\}$, while customers $\{0,3,6,7,9\}$ are assigned to depot $\{1\}$. The two customer sequences will directly be input for LS-VND afterwards.

4.2.2 Child generation

The process of generating a child is as follows, two parents P_1 and P_2 are selected from the original population with uniform probability, and then undergo crossover procedure that generates a child C. The crossover procedure which we proposed is not complex, for each depot, the first step is to inherit data from P_1, and then inherit data from P_2. Finally, we assign to the remaining customers. The framework and examples of this procedure are given below.

Algorithm 4: Crossover procedure

Step 1: inherit data from parents

1(a): for each depot d in $|D|$, generate a random number from 0 to $nc_d(P_1)$ denoted by n_1, generate another random number from 0 to $nc_d(P_2)$ denoted by n_2;

1(b): inherit data from P_1: copy the first n_1 customers from P_1 to offspring C;

1(c): inherit data from P_2: copy the last $nc_d(P_2) - n_2$ customers, except those been copied from P_1, from P_2 to C;

Step 2: complete customer assignment

2(a): for all customers that have not been assigned to the depot, randomly select a depot $\bar{d}$, add the customer to the customer sequence in the depot $\bar{d}$ in C.

The algorithm of crossover procedure in generating a child is shown in Algorithm 4, $nc_d(P_1)$ is the number of customers assigned to depot d in P_1. To facilitate the interpretation, an instance is given as shown in Figure 7.

As shown in Figure 7, first we take depot $\{0\}$ into account, customer $\{2\}$ is copied from P_1 to C, and customer $\{1,0\}$ are inherited from P_2 to C. After all depots are considered, the remaining customers are $\{6,8\}$, and then we conduct the next step. For each of them, we randomly choose a depot, and add the customer to the customer sequenceof the depot in offspring C, thus the offspring C is completed, as shown in Figure 7(b).

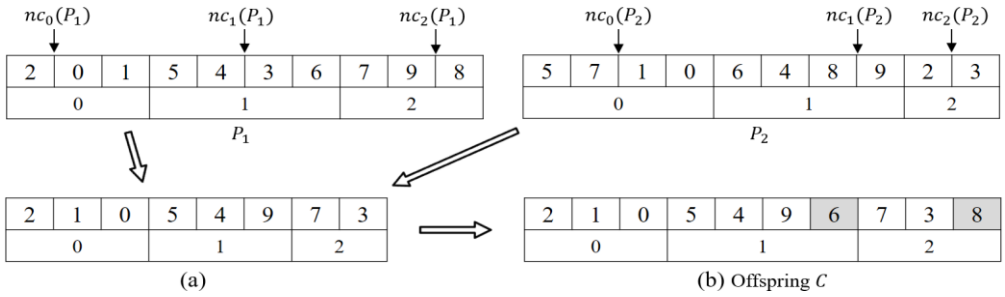


Figure 7: Child generating in HGA

4.2.3 Fitness evaluation in HGA

In general, the cost function (i.e., fitness function) is utilized as the evaluation function to perform survivor selection. However, it usually leads to undesirable approximation due to the neglect of potential outstanding individuals that may have a better diversity of the population. Here, an evaluation function is proposed to contain both the cost and its diversity contribution to the population of an individual.

We account the diversity contribution $R_{dc}(P)$ as the average distance for an individual P to its n_c closet individuals, grouped in a set N_C, the distance $dis(P_1, P_2)$ between two individual P_1 and P_2 is the number for customers that assigned to different depots, called *broken* (Prins, 2009), formulated as follows.

$$R_{dc}(P_1) = \frac{1}{n_c}\sum_{P_2} dis(P_1, P_2) \tag{29}$$

$$dis(P_1, P_2) = \frac{1}{|N|}\sum_{i \in N} 1(dep_i(P_1) \neq dep_i(P_2)) \tag{30}$$

$dep_i(P_1)$ is the assigned depot to serve customer i in individual P_1, and '$1(condition)$' equals 1 if the '*condition*' is true, and 0 otherwise. A *biased fitness* function $R_{bf}(P)$ is the proposed evaluation function of individual P, let $R(P)$ be the penalized cost of individual P. $R(P)$ can be obtained through labeling procedure via the function (22) as discused in Section 4.1.3, thus

$$R_{bf}(P) = R(P) - (1 - \frac{\pi}{\pi + \mu})R_{dc}(P) \tag{31}$$

The biased fitness function will be utilized in survivor selection later. Note that the value of $R_{bf}(P)$ should mainly depend on $R(P)$. Here the normalization of $R(P)$ and $R_{dc}(P)$ is not performed because $R(P)$ is much larger than $R_{dc}(P)$ based on the instances in the benchmark.

4.2.4 Reorder routine in HGA

In HGA, we adopt a reorder routine to reconstruct individuals after LS-VND. The reorder routine can be described as follows. For each individual, its customer sequence will be reorded according to the best sequence found by LS-VND. It also can be seen as an etilism strategy to accelerate the algorithm, and each individual is reconstructed with an elite customer sequence before it enters the next generation. The idea of reorder routine is similar to solution translation in HPSO, and both of them act on accelerating algorithm by keeping the best representations that can be obtained from individuals. They are regarded as acceleration modules that we designed for HPSO and HGA.

4.2.5 Survivor selection

The role of the survivor selection procedure is to preserve relatively better solutions and contribute to the diversity of the population. Besides, we name the individual P_2 that have the same assignments of customers as another individual, which means $dis(P_1, P_2) = 0,$ and the copy can be discarded. The framework of a survivor selection procedure is shown as follows.

Algorithm 5: Survivor selection procedure

1: **for** i equals 0 to $\pi - 1$
2: add all *Copy* to a set G
3: **if** $G \neq \emptyset$,
remove individual $P \in G$ which has the maximum biased fitness
4: **else**
remove P in the population which has the maximum biased fitness
5: update the value of distance and biased fitness
6: **end**

It should be noted that in the proposed survivor selection procedure if an individual P_1 has only one copy (say P_2), then both P_1 and P_2 are copies, but only one of them will be discarded with the update mechanism in the distance and biased fitness. The proposed procedure will discard, first copy with bad biased fitness, then the bad biased fitness that not belong to copy.

5. Numerical experiments

In this section, we conduct numerical experiments using a PC (Intel Core i7, 2.80 GHz, 8 GB of Memory) to validate the effectiveness of our proposed models and the efficiency of our developed solution methods. The model is implemented by CPLEX (Version 12.6.2) with concert technology of C# (VS2015). Note to tackle the nonlinear form *max/min* in our model, CPLEX can be coded as 'model.Max(a,b)/model.Min(a,b)', where 'a' and 'b' are the two comparison terms. Parameter settings are reported in Section 5.1. Section 5.2 reveals the performance of the heuristics. Experiments on algorithm settings are listed in Section 5.3. In addition, two experiments are conducted to verify the applicability of the algorithms on two variants of our problem, as reported in Appendix 4.

5.1 Parameters setting

Our numerical experiments are performed on small-scale instances and large-scale instances. Some tests on algorithm settings are also conducted. The parameters of customer locations, vehicle capacity, and the time windows are based on Solomon's benchmark (Solomon, 1987) and are selected randomly from its data set C101. The locations of depots are randomly generated in a two-dimensional coordinate area where the customers are located at. As for the release date parameter settings, we use the method proposed by Cattaruzza et al., (2016) and the program is outlined in Appendix 2.

In the preliminary tests, the value of ϑ_{thr} in labelling procedure has a significant impact on the solution, as shown in Appendix 3. This is due to that ϑ_{thr} has an affection on γ (see formula (26)) while γ has a significant impact on labelling procedure as a parameter in elimination mechanism.

After previous numerical tests, the parameters are set as follows. θ equals 10 and λ equals 2, and ϑ_{thr} equals 10 as in labeling procedure; w equals 1.3, c_1 and c_2 equal 1, and the size of particles in the swam is 25 in HPSO; the number of elite individuals, i.e., π, equals 20, μ equals 25, and n_c equals 10 in HGA. The maximum number of iteration in HPSO and HGA are both set to 300, and if the best solution does not get improve for 30 consecutive iterations, then stop iteration and output the current best solution.

5.2 Performance of the HPSO and the HGA

The results of comparing HPSO and HGA with CPLEX are illustrated in Tables 1 and 2, where the objective values 'Z', the computation time 'T', and $\text{Time ratio} = (T_p/T_c)$ or (T_g/T_c) are provided. Here, Z_c is the optimal solution obtained by CPLEX; Z_p and Z_g are the solution obtained by the HPSO and the HGA, respectively; T_c is the CPLEX computation time in seconds; T_p and T_g are the HPSO's and the HGA's computation time in seconds.

Table 1: Performance of the HPSO and the HGA for small-scale instances

Instance ID	CPLEX		HPSO		Time ratio	HGA		Time ratio
	Z_c	$T_c(s)$	Z_p	$T_p(s)$	T_p/T_c	Z_g	$T_g(s)$	T_g/T_c

A5-2-4-3-1	52.0	1.6	52.0	0.2	0.13	52.0	0.2	0.13
A5-2-4-3-2	58.7	1.6	58.7	0.2	0.13	58.7	0.2	0.13
A5-2-4-3-3	54.6	1.6	54.6	0.2	0.13	54.6	0.2	0.13
A10-2-4-3-1	223.1	4.5	223.1	1.0	0.22	223.1	1.0	0.22
A10-2-4-3-2	148.2	5.1	148.2	1.0	0.20	148.2	1.1	0.22
A10-2-4-3-3	173.5	4.7	173.5	1.0	0.21	173.5	1.1	0.23
A15-2-4-3-1	279.3	31.1	279.3	6.8	0.22	279.3	7.9	0.25
A15-2-4-3-2	327.1	23.8	327.1	6.5	0.27	327.1	6.7	0.28
A15-2-4-3-3	312.2	45.8	312.2	7.1	0.16	312.2	6.9	0.15
A20-2-4-3-1	380.7	342.7	380.7	39.7	0.12	380.7	45.1	0.13
A20-2-4-3-2	368.7	633.0	368.7	45.4	0.07	368.7	44.4	0.07
A20-2-4-3-3	398.6	587.9	398.6	41.4	0.07	398.6	43.4	0.07
A25-2-4-3-1	434.7	1806.9	434.7	53.1	0.03	434.7	53.2	0.03
A25-2-4-3-2	421.8	3765.2	421.8	56.9	0.02	421.8	54.9	0.01
A25-2-4-3-3	359.7	3581.6	359.7	52.2	0.01	359.7	55.9	0.02
Avg.		**722.5**		**20.8**	**0.03**		**21.5**	**0.03**

We tested a total of 15 instances for 5, 10, 15, 20, and 25 customers as shown in Table 1, each instance ID contains five parts, they are the number of customers, the number of depots, the number of vehicles, the maximum number of trips that each car can perform, and the instance index in turn. When the number of customers in the instances is up to '30', the proposed model becomes intractable for CPLEX to solve directly (cannot get a feasible solution within three hours). In terms of small-scale instances, the HPSO (resp., the HGA) can also obtain the optimal solutions, but it only takes an average of 20.8 (resp., 21.5) seconds of computation time. On large-scale instances where CPLEX cannot obtain solutions, the proposed methods can provide feasible solutions in a reasonable time as shown in Table 2.

We list a total of 12 instances from 30 to 90 customers for testing large-scale instances, where each instance ID means the same as in *Table 1*. The average solution value of HPSO (resp., the HGA) is 1046.5 (resp., 1036.1), and the average solution time is 554 (resp., 529). The results show that the average gap between HPSO and HGA is 0.55%, and the average time ratio is 0.97. HGA performs slightly better than HPSO.

Table 2: Performance of the HPSO and the HGA for large-scale instances

Instance ID	HPSO		HGA		Gap(%)	Time ratio
	Z_p	$T_p(s)$	Z_g	$T_g(s)$	$(Z_p - Z_g)/Z_p$	T_g/T_p
A30-3-6-3-1	426.4	198	424.5	213	0.45	1.08

A30-3-6-3-2	323.2	254	323.2	259	0	1.02
A35-4-8-3-1	332.2	321	330.8	323	0.42	1.01
A35-4-8-3-2	396.7	317	395.8	307	0.23	0.97
A40-4-8-3-1	616.3	457	615.4	412	0.15	0.9
A40-4-8-3-2	482.8	460	480.9	431	0.39	0.94
A45-5-10-3	670.5	575	668.7	536	0.27	0.93
A50-5-10-3	738.6	645	732.7	607	0.8	0.94
A60-6-12-3	1186.8	626	1172.7	601	1.19	0.96
A70-7-14-3	1517	778	1514.9	722	0.14	0.93
A80-8-16-3	2100.7	866	2096.3	851	0.21	0.98
A90-9-18-5	3767.4	1152	3678.2	1087	2.37	0.94
Avg.	**1046.5**	**554**	**1036.1**	**529**	**0.55**	**0.97**

Since the number of customers in Solomon's benchmark is no more than one hundred, we adopt the Homberger's benchmark (customers are distributed in a larger space than Solomon's benchmark) to test our algorithms for the larger scale instances with customers arranged from 100 to 200. We list 11 instances from 100 to 200 customers, as shown in Table 3, that each instance ID contains four parts, i.e., number of customers, number of depots, number of vehicles and the maximum number of trips for each vehicle. In very-large-scale instances, we can see that HGA is better than HPSO on both objective value and solution time.

To sum up, the above observation indicates that the HPSO and the HGA can provide good solution effectively and have a significant advantage in computation time. In small-scale instances, both of HPSO and HGA can obtain the optimal solution in a short time (within in one minute), but the computation time of CPLEX solver has reached one hour, in addition, HPSO is slightly faster than HGA. In large-scale instances, CPLEX could not obtain the solution in a reasonable time, while HPSO and HGA still have the ability to solve the problem in a relatively short time, but HGA has the better performance than HPSO. Therefore, HGA is more suitable for large-scale instances than HPSO.

Table 3: Performance of the HPSO and the HGA for very-large-scale instances

Instance ID	HPSO		HGA		Gap(%)	Time ratio
	Z_p	$T_p(s)$	Z_g	$T_g(s)$	$(Z_p - Z_g)/Z_p$	T_g/T_p
A100-10-20-5	4512.6	1269.5	4489.4	1210.4	0.51	0.95
A110-11-22-5	5244.2	1281.7	5231.0	1245.7	0.02	0.97
A120-12-24-5	5572.6	1522.6	5537.9	1498.3	0.62	0.98
A130-13-26-5	5789.3	1756.2	5768.1	1704.4	0.37	0.97
A140-14-28-5	6259.2	2522.6	6208.3	2401.7	0.81	0.95
A150-15-30-5	6628.6	3906.3	6632.7	3789.7	-0.06	0.97

A160-16-32-5	7620.3	5469.2	7515.9	5532.6	1.37	1.01
A170-17-35-5	8243.6	5481.3	8199.7	5379.6	0.53	0.98
A180-18-36-5	8406.8	6963.3	8404.3	6896.5	0.03	0.99
A190-19-38-5	14240.	7869.6	14238.4	7787.9	0.02	0.99
A200-20-40-5	16138.	8361.7	16107.8	8203.5	0.19	0.98
Avg.	**6488.6**	**3319.8**	**6463.7**	**3263.3**	**0.38**	**0.97**

We also conduct the experiments by comparing the costs of the two algorithms with the same time limit, and the results of 18 instances for 10 to 80 customers are listed as shown in Table 4. In each instance, the time limit is based on the earlier stop time of iteration from the two algorithms. The results indicate the same conclusion as the experiments above. Here we can conclude that HGA performs better than HPSO in large-scale instances.

Table 4: Comparison between the HPSO and the HGA with the same time limit

Instance ID	Z_{HPSO}	Z_{HGA}	T(s)
B10-2-4-3-1	**98.5**	100	1.2
B10-2-4-3-2	134.2	134,2	1.1
B10-2-4-3-3	213	**204.1**	1.6
B20-2-4-3-1	**286.2**	300.7	71
B20-2-4-3-2	321.6	**283.2**	41
B20-2-4-3-3	**298.3**	304.1	28.4
B30-3-6-3-1	427.4	**291.9**	421
B30-3-6-3-2	587.4	**360.4**	551
B30-3-6-3-3	526.9	**475**	639
B40-4-8-3-1	948.4	**770.2**	554
B40-4-8-3-2	1055.8	**995.4**	296
B40-4-8-3-3	968.6	**602.7**	981
B60-6-12-3-1	3050.1	**1627**	949
B60-6-12-3-2	2517.3	**1822.5**	806
B60-6-12-3-3	3051.1	**1748.7**	534
B80-8-16-3-1	5142.5	**3482.2**	1236
B80-8-16-3-2	6514	**4690**	970
B80-8-16-3-3	5013.5	**3750.1**	621

In addition, we perform the experiments by taking different numbers of depots and vehicles into consideration regarding the same number of customers, we list 16 instances for 10 to 70 customers, as shown in Table 5, the instance ID denotes the number of customers, number of depots, number of all vehicles and number of trips. We do not list the tests on the maximum number of trips for each vehicle due to that the results show no clear differences.

Table 5: Additional experiments on the number of depots and vehicles

Instance ID	HPSO		HGA		Gap(%)	Time ratio
	Z_p	$T_p(s)$	Z_g	$T_g(s)$	$(Z_p - Z_g)/Z_p$	T_g/T_p
C10-2-4-3	213	1.1	213	1.6	0	1.45
C10-2-6-3	134.2	0.9	134.2	1.2	0	1.33
C10-3-6-3	163.6	3.4	163.6	1.8	0	0.53
C10-3-9-3	163.6	1.4	163.6	2.2	0	1.57
C30-2-6-3	432.5	536	429.5	589	0.69	1.1
C30-3-6-3	391.7	366	385.4	429	1.61	1.17
C30-3-9-3	362.3	352	357.5	347	1.32	0.99
C30-4-8-3	443.3	384	438.1	349	1.17	0.91
C50-3-9-3	1124.5	1317	1094.4	1226	2.68	0.93
C50-4-12-3	1023.9	1051	989.4	989	3.37	0.94
C50-5-10-3	1473.2	725	1450	635	1.57	0.88
C50-7-14-3	1045.6	763	1018.9	824	2.55	1.08
C70-4-12-3	2850.5	919	2697.3	873	5.37	0.95
C70-6-12-3	2417.5	1043	2302.6	979	4.75	0.94
C70-8-16-3	2428.1	723	2344.1	844	3.46	1.17
C70-10-20-3	2742.9	1447	2575.6	1311	6.1	0.91
Avg.	**1088.2**	**602.1**	**1047.3**	**587.6**	**2.16**	**1.05**

Some interesting results are shown in Table 5. On the one hand, the costs decrease as the number of vehicles increases (see instances *C10-* and *C30-*). On the other hand, increasing the number of warehouses while the total number of vehicles unchanged does not necessarily reduce the costs, a possible explanation is that with the reduction of the average number of vehicles for each depot, some customers must be assigned to a farther depot which may not suitable for the delivery. Thus, another conclusion can be drawn for the delivery companies: With relatively lower distribution costs, choosing suitable locations for depots can help the companies reduce the fixed costs (e.g., the construction cost of depots and acquisition cost of vehicles). This relates the Facility Location Problem and we do not give more discussion here.

5.3 Experiments on algorithm settings

We conduct two comparative experiments to compare our proposed algorithms and the simplified algorithms without either the module of LS-VND (section 4.1.4) or the module of acceleration (section 4.1.5 and 4.2.4), respectively. The module of acceleration corresponds to the solution translation mechanism in HPSO and the reorder routine in HGA. The data is also randomly selected from the benchmark. The first comparative experiment is to investigate the efficiency of the VND module. The results are shown in Table 6 and Figure 8, which are the

comparison related to the HPSO and the HGA, respectively. The second comparative experiment is to investigate the efficiency of the acceleration module. The results are shown in Figure 9 and Figure 10, which are the comparison related to the HPSO and HGA, respectively.

We list 5 experiments on 20 customers, as shown in Table 6. Z_{nv} and T_{nv} are the solutions and time of the simplified algorithm without VND module. The average gap of the solution obtained by the HPSO without VND module is 53.68%, and the average time ratio is 1.51. Additional experiments for HPSO on 70 and 120 customers without VND module cannot find a feasible solution, thus are not listed here.

Table 6: Comparison of HPSO and HPSO (Without VND)

Instance ID	HPSO		HPSO(Without VND)		Gap	Time ratio
	Z_p	T_p	Z_{nv}	$T_{nv}(s)$	$(Z_{nv}-Z_p)/Z_p$	T_{nv}/T_p
D20-2-4-3-1	306.6	27.9	497.8	37.59	62.38%	1.35
D20-2-4-3-2	283.2	28.7	457.6	40.05	61.60%	1.40
D20-2-4-3-3	275.8	25.3	428.0	45.73	55.17%	1.81
D20-2-4-3-4	235.9	31.6	337.3	42.44	42.98%	1.34
D20-2-4-3-5	237.1	42.7	346.8	69.63	46.29%	1.63
Avg.					**53.68%**	**1.51**

We list 15 experiments on scales of 20, 70 and 120 customers as Figure 8 illustrates, for the average results on each scale of the simplified HGA without VND module in comparison with to HGA. Z_{nv} and T_{nv} denote the solution objective value and solution time of HGA without VND module, separately. The results concluded that HGA without VND module has an average gap of 27.58% and an average time ratio of 1.31. However, HGA can still solve 120 customers after removing VND module. The results of Table 6 and Figure 8 validate the efficiency of VND module. VND module can improve the quality of the solution.

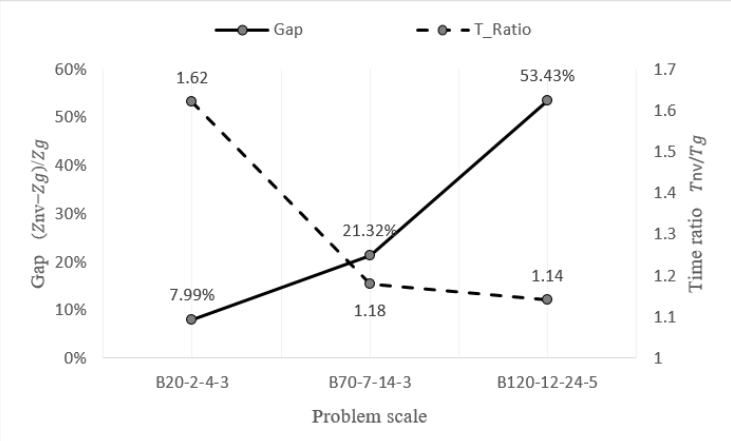


Figure 8: Comparison of HGA and HGA (Without VND)

We list 15 experiments on scales of 20, 70 and 120 customers as illustrated in Figure 9, Z_{ns} and T_{ns} denote the solution value and solution time of HPSO without solution translation module, respectively. The average results of the HPSO without solution translation mechanism compare to the HPSO in each scale. The average gap is 4.85% and the average time ratio is 5.23.

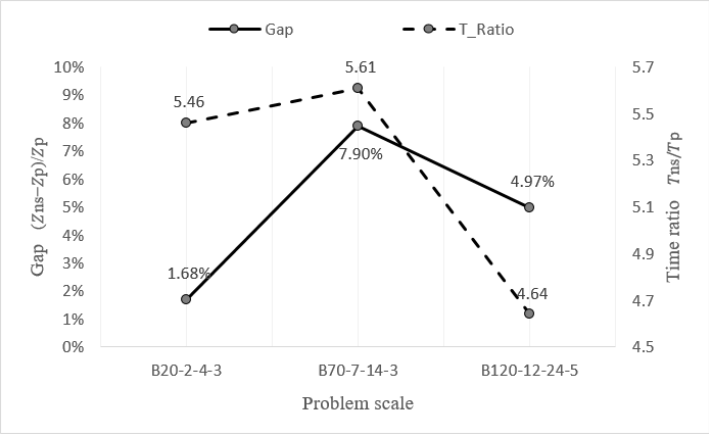


Figure 9: Comparison of HPSO and HPSO (Without solution translation (S.T.))

We list 15 experiments on the scales of 20, 70 and 120 customers as shown in Figure 10, for the results of the simplified HGA without reorder routine, it has an average gap of 2.52% and an average time ratio of 5.16 compared to HGA. The results of Figure 9 and Figure 10 demonstrate that the role of acceleration module is to increase the speed of the algorithm.

In summary, the VND module and the acceleration module in our proposed algorithms have significant benefits on solution quality and computation performance.

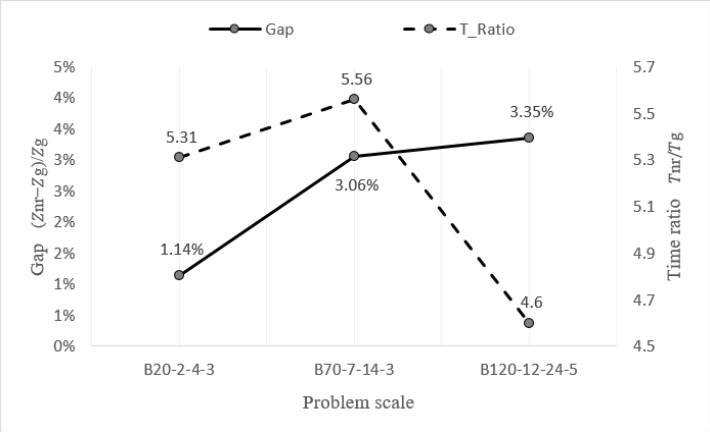


Figure 10: Comparison of HGA and HGA (Without reorder routine (R.R.))

6. Conclusions

In the last mile distribution networks, travel time for delivering packages from depots to customers depends on the trips and schedules for a fleet of vehicles serving customers. We study the multi-depot multi-trip vehicle routing problem with time windows and release dates, which optimizes the assignment of trips and customers to vehicles as well as the sequence for vehicles visiting customers. The contributions of this study are listed as follows: first, this paper makes an explorative study on a variant of vehicle routing problem by considering multiple depots, multiple trips, time windows and release dates, which are seldom considered simultaneously in existing studies. Moreover, the multi-depot multi-trip vehicle routing problem with time windows and release dates is formulated as a mixed integer programming model to minimize the total travel time of vehicles with satisfying the time windows, the release dates requested by customers, and vehicle capacity constraints. For solving the proposed model in a fast way, we design a hybrid particle swarm optimization algorithm (HPSO) and a hybrid genetic algorithm (HGA) into which a solution translation and reorder routine are respectively involved. Last, we conduct numerical experiments to test the efficiency of the two solution methods.

The experimental results show that our proposed algorithms not only obtain the optimal solution for some small-scale problem instances but also solves some large-scale instances (200 customers, 20 depots and 40 vehicles) in reasonable computation time. The two proposed meta-

heuristics have a similar ability and the minor differences can be summarized that HGA has better applicability for large scale. In addition, the acceleration mechanisms, i.e., solution translation in HPSO and reorder routine in HGA, have a significant improvement in solution quality and computing time.

References

Alvarez, A., Munari, P. (2017). An exact hybrid method for the vehicle routing problem with time windows and multiple deliverymen. *Computers & Operations Research* 83: 1−12.

Archetti, C., Feillet, D., Mor, A., & Speranza, M. G. (2018). An iterated local search for the Traveling Salesman Problem with release dates and completion time minimization. *Computers & Operations Research* 98: 24−37.

Archetti, C., Feillet, D., Speranza, M. G. (2015). Complexity of routing problems with release dates. *European Journal of Operational Research* 247(3): 797−803.

Azadeh, A., Elahi, S., Farahani, M. H., Nasirian, B. (2017). A genetic algorithm-Taguchi based approach to inventory routing problem of a single perishable product with transshipment. *Computers & Industrial Engineering* 104: 124−133.

Bae, H., Moon, I. (2016). Multi-depot vehicle routing problem with time windows considering delivery and installation vehicles. *Applied Mathematical Modelling* 40(13 − 14): 6536−6549.

Bent, R., Hentenryck, P. V. (2004). A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science* 38(4): 515−530.

Bettinelli, A., Ceselli, A., Righini, G. (2011). A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies* 19(5): 723−740.

Cattaruzza, D., Absi, N., Feillet, D. (2016). The Multi-Trip Vehicle Routing Problem with Time Windows and Release Dates. *Transportation Science* 50(2): 676−693.

Dantzig, G. B., Ramser, J. H. (1959). The truck dispatching problem. *Management Science* 6(1): 80−91.

Desrochers, M., Desrosiers, J., Solomon, M. M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research* 40(40), 342−354.

Desaulniers, G., Lavigne, J., Soumis, F. (1998). Multi-depot vehicle scheduling problems with time windows and waiting costs. *European Journal of Operational Research* 111(3): 479−494.

Diego Muñoz-Carpintero, Doris Sáez, Cristián E. Cortés, Alfredo Núñez (2015). A Methodology Based on Evolutionary Algorithms to Solve a Dynamic Pickup and Delivery Problem Under a Hybrid Predictive Control Approach. *Transportation Science* 49(2): 239−253.

Dondo, R. G., Cerdá, J. (2009). A hybrid local improvement algorithm for large-scale multi-depot vehicle routing problems with time windows. *Computers & Chemical Engineering* 33(2): 513−530.

Eberhart R., Kennedy J. (1995). A new optimizer using particle swarm theory. *International Symposium on Micro Machine & Human Science* 2002: 39−43.

Hernandez, F., Feillet, D., Giroudeau, R., Naudd, O. (2016). Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. *European Journal of Operational Research* 249(2): 551−559.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science* 43(4): 408−416.

Nagata Y., Bräysy O., Dullaert W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research* 37(4): 724−737.

Nagy, G., Wassan, N. A., Speranza, M. G., & Archetti, C. (2015). The vehicle routing problem with divisible deliveries and pickups. *Transportation Science* 49(2): 271−294.

Nguyen, P. K., Crainic, T. G., Toulouse, M. (2013). A tabu search for time-dependent multi-zone multi-trip vehicle routing problem with time windows. *European Journal of Operational Research* 231(1): 43−56.

Pinedo, M. (2005). *Planning and scheduling in manufacturing and services*. Springer (New York).

Potvin, J. Y. (2009). State-of-the art review---evolutionary algorithms for vehicle routing. *Informs Journal on Computing* 21(4): 518−548.

Prins C. (2009). Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence* 22(6): 916−928.

Reyes, D., Erera, A. L., Savelsbergh, M. W. P. (2018). Complexity of routing problems with release dates and deadlines. *European Journal of Operational Research* 266(1): 29−34.

Salhi, S., Imran, A., Wassan, N. A. (2013). The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighborhood search implementation. *Computers & Operations Research* 52(PB): 315−325.

Shelbourne, B. C., Battarra, M., & Potts, C. N. (2017). The vehicle routing problem with release and due dates. *INFORMS Journal on Computing* 29(4): 705−723.

Solomon M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research* 35(2): 254−265.

State Post Bureau of The People's Republic of China, (2018). China express development index from 2016 to 2018. http://www.spb.gov.cn/sj/zgkdfzzs/. Accessed by 13 April 2018.

Tang, J. , Yu, Y. , & Li, J. . (2015). An exact algorithm for the multi-trip vehicle routing and scheduling problem of pickup and delivery of customers to the airport. *Transportation Research Part E* 73: 114−132.

Toth P., Vigo D., Toth P. (2014). *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. Society for Industrial and Applied Mathematics.

Vidal T., Crainic T. G., Gendreau M. (2012). A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems. *Operations Research* 60(3): 611−624.

Wassan, N., Wassan, N. A., Salhi, S., Nagy, G. (2017). The Multiple Trip Vehicle Routing Problem with Backhauls: Formulation and a Two-Level Variable Neighbourhood Search. *Computers & Operations Research* 78(C): 454−467.

Yan, S., Chu, J. C., Hsiao, F. Y., Huang, H. J. (2015). A planning model and solution algorithm for multi-trip split-delivery vehicle routing and scheduling problems with time windows. *Computers & Industrial Engineering* 87: 383−393.

Appendix

Appendix 1. An example of a labeling procedure

If a customer sequence $\{c_1, c_2, c_3\}$ is assigned to depot d, and it has two homogeneous vehicles, k_0 and k_1, with a capacity equal 200 while the maximum number of trips equals 2 for each vehicle, the information of the customers and the depot is shown in Table Ap.1(a) and a distance matrix in Table Ap.1(b) below. For $\theta = 5, \lambda = 5, \vartheta_{thr} = 10$ and γ initially set to 1, the labels correspond to each customer node will be produced, as shown in Table Ap.1(c).

Table Ap.1(a): Information

	t^S	q_i	e_i	l_i	t^R
$\boldsymbol{d}$	20	0	0	200	0
$\boldsymbol{c_1}$	5	20	100	120	60
$\boldsymbol{c_2}$	5	20	50	100	20
$\boldsymbol{c_3}$	5	20	130	160	60

Table Ap.1(b): Distance matrix

	d	c_1	c_2	c_3
$\boldsymbol{d}$	0	5	10	15
$\boldsymbol{c_1}$	5	0	10	20
$\boldsymbol{c_2}$	10	10	0	30
$\boldsymbol{c_3}$	15	20	30	0

Table Ap.1(c): Generation of Labels correspond to customer sequence

$\boldsymbol{c_1}$					$\boldsymbol{c_2}$					$\boldsymbol{c_3}$				
Time	*P*	*Cost*	*Dom*	*Fesi*	*Time*	*P*	*Cost*	*Dom*	*Fesi*	*Time*	*P*	*Cost*	*Dom*	*Fesi*
(110,0)	d	10	*N*	*Y*	(115,0)	d	100	*N*	*N*	(155,0)	d	135	*N*	*N*
					(115,0)	c_1	230	*Y*	*N*	(155,0)	c_1	265	*Y*	*N*
					(110,65)	c_1	30	*N*	*Y*	(150,110)	c_1	65	*Y*	*Y*
										(170,0)	c_2	130	*N*	*N*
										(150,115)	c_2	130	*Y*	*N*
										(165,65)	c_2	60	*N*	*Y*
										(150,110)	c_2	60	*N*	*Y*

Note: In the columns of '*Dom*' and '*Fesi'* elements, 'N' means 'No', 'Y' means 'Yes'.

For element *P*, when generating a label for a customer c_i $(i = 1,2,3)$ in customer sequence $\{c_1, c_2, c_3\}$, regarding c_i as the *considering customer*, assuming the trip that contains the *considering customer* (i.e., c_i) as the *state trip*, then the customer which is just before the first

customer of the *state trip* in $\{c_1, c_2, c_3\}$ is regarded as the predecessor node as well as element *P* of the label.

For c_1 there is only one label since only one situation exists, i.e., a vehicle (denoted as k_0 as well) travels from depot d to customer c_1 and returns to d after cargos discharged at c_1. Thus, the corresponding route for k_0 is $\{d \to c_1 \to d\}$ (the *state trip* as well) and the information is retained. The representation of each event node on the timeline is shown in the Figure Ap.1. The vehicle k_0 departs from depot d at time 80 (i.e., $t_{c_1}^R + t^D$ as $t_{c_1}^R$ equals 60 and t^D equals 20 in Table Ap.1(a)) and arrives at c_1 at time 85, it starts to serve c_1 at time 100 due to the time window and ends the service at time 105, at the same time the vehicle begins to return to depot and will arrive at the depot at time 110. Thus, the element *Cost* only contains the vehicle traveling distance that equals 10 without penalty. In the customer sequence (i.e., $\{c_1, c_2, c_3\}$), since there is no customer before the *state trip*, element *P* is denoted as the depot d. The ready *Time* for the two vehicle is (110, 0). The element *e* is *Y* as the arrangement is feasible.

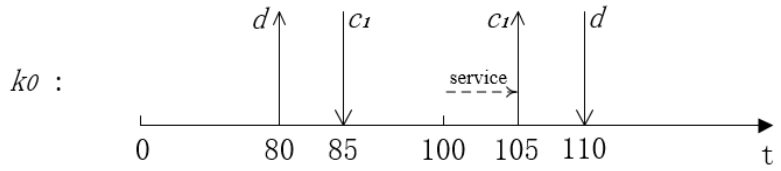


Figure Ap.1: Event nodes of vehicles on the timeline for the label of c_1

Then for c_2 there would be more labels as the situation varies. The three labels corresponded to c_2 in Table Ap.1(c) will be explained in detail below.

For the first label of c_2, the two customers c_1 and c_2 are assigned to the same vehicle (denoted as k_0 as well) and the vehicle serves c_1 and c_2 in turn. Thus the route (the first trip as well the *state trip*) of k_0 is $\{d \to c_1 \to c_2 \to d\}$. The corresponding representation is shown in the Figure Ap.2. Vehicle k_0 departs from depot d at time 80 and arrives at c_1 at time 85. It starts to serve c_1 at time 100, ends the service at time 105 and at this moment k_0 starts to go to c_2, by the time at 115 k_0 arrives at c_2. Note that the time window for the start time of service for c_2 is $[50, 100]$, thus it starts to serve c_2 at time 100 and ends the service at time 105, meanwhile k_0 starts to go back to depot d as arrives at the depot at time 115.

Therefore, the element *Cost* (equals 100) contains two parts, i.e., the traveling distance of vehicle (i.e., 25) and a penalty from violation of time window (i.e., 75, with time violation of 15). Due to there is no customer before the *state trip* in the customer sequence, element *P* is noted as d. The ready *Time* for the two vehicle is (110, 0). The element *Fesi* is *N* since there is a violation in time window.

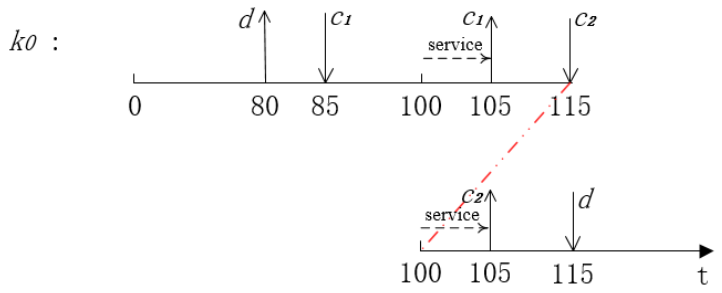


Figure Ap.2: Event nodes of vehicles on the timeline for the first label of c_2

For the second label of c_2, the situation changes as follows: customer c_1 and c_2 are assigned to the same vehicle (denoted as k_0 as well) but in different trips. So k_0 performs two trips, i.e., the first trip is $\{d \to c_1 \to d\}$ while the second trip (the *state trip* as well) is $\{d \to c_2 \to d\}$. The corresponding representation is shown in the Figure Ap.3. The *Cost* equals 230 which contains two parts: traveling distance of vehicle (i.e., 30) and a penalty from violation of time window (i.e., 200, with time violation of 40). The element *Time* is (115, 0). Since the predecessor node before the *state trip* (i.e., the second trip of k_0) is c_1 in the customer sequence, element *P* is noted as c_1.

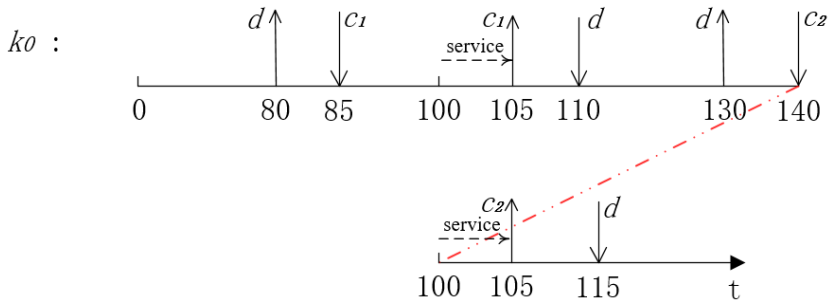


Figure Ap.3: Event nodes of vehicles on the timeline for the second label of c_2

For the third label of c_2, the situation is different from the first two labels above as customer c_1 and c_2 are assigned to different vehicles (k_0 and k_1), the routes of the two vehicles are as follows: k_0 performs one trip $\{d \to c_1 \to d\}$ and k_1 performs another trip $\{d \to c_2 \to$

d}. The corresponding representation is shown in the Figure Ap.4. The element *Cost* is 30, element *P* is c_1, *Time* is (110, 65) and *Fesi* is Y as there is no violation on neither time window nor vehicle load.

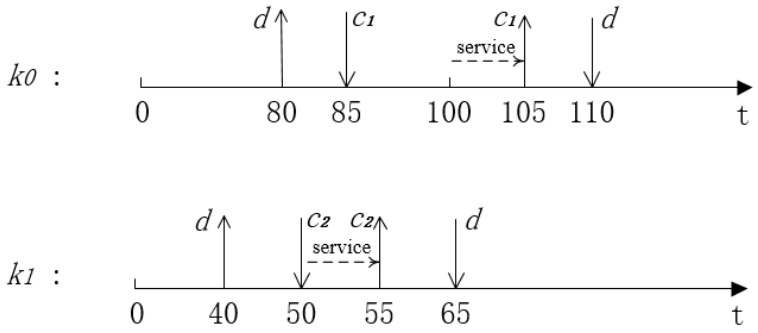


Figure Ap.4: Event nodes of vehicles on the timeline for the third label of c_2

There is no more different situation of arrangement if the vehicles are homogeneous, thus dominance rule will be implemented for the three labels to obtain the element *Dom* of these labels. All the labels with *Y* of the element *Dom* are eliminated. As a consequence, only the first label and the third label of c_2 are reserved.

It should be noted that in the second and the third label of c_2, the element *P* are the same, denoted as c_1, and it can be regarded that these two labels are generated based on the non-dominated labels of c_1. In specific, since there is only one label as is not dominated in c_1, it can be considered that the generation of the second and the third labels in c_2 are obtained by assigning the operating customer (i.e., c_2) to a different vehicle to form a new trip.

For the last customer node c_3 in the customer sequence, the situations of the labels' generation are the same as described in the customer c_2, and thus no repeat discussions are given here. There are two labels with the same lowest cost (i.e., 60) while both are feasible, then the former label is selected, and the corresponding assignments of the trips are as follows: k_0 performs two trips, $\{d \to c_1 \to d,\ d \to c_3 \to d\}$, and k_1 performs one trip, $\{d \to c_2 \to d\}$.

Appendix 2. Release dates generating

set $\boldsymbol{\Phi} = \emptyset;\ \boldsymbol{\Omega} = \emptyset;\ \boldsymbol{\Delta} = \infty;\ \boldsymbol{R} = \mathbf{0}$
for i equals 1 to $|N|$
  $t_i^R \leftarrow max\ (0, \lfloor L_i - 0.5T^H \rfloor)$
  if $t_i^R > L_i - \max\limits_{d \in D_k} t_{d,i}^P$
    $t_i^R \leftarrow 0$
  if $t_i^R > 0$
    $\Phi \leftarrow \Phi \cup c_i$
end for
sort Φ with customers by non-decreasing release dates: $t_i^R \le t_j^R \Rightarrow c_i < c_j$
while $\Phi \neq \emptyset$
  $c^* \leftarrow$first customer in Φ
  $\Phi \leftarrow \Phi \backslash \{c^*\}$
  if $\Omega = \emptyset$
    $\Omega \leftarrow \{c^*\};\ \Delta \leftarrow \left(\varpi L_{c^*} - \max\limits_{d \in D} t_{d,c^*}\right);\ R \leftarrow t_{c^*}^R$
  else
    $\Delta \leftarrow \min\{\Delta, \varpi L_{c^*} - \max\limits_{d \in D} t_{d,c^*}\}$
    if $t_{c^*}^R \le \Delta$
      $\Omega \leftarrow \Omega \cup \{c^*\};\ R \leftarrow t_{c^*}^R$
    else
      $\Phi \leftarrow \Phi \cup \{c^*\}$
      for all $c \in \Phi$
        $t_c^R \leftarrow R$
      end for
      $\Omega \leftarrow \emptyset$
end while

We set ϖ be 0.75 according to the reference, ϖ is a parameter that can influence the rigidity in release date generating, the reference gives four values of ϖ, $\{0, 0.25, 0.5, 0.75\}$ in testing and 0.75 is finally suggested. The idea of the release dates generating program is to generate release date for customers firstly based on its time window and the depots service time to avoid the possible situation that no feasible solution exists, secondly cluster the customers with the release dates generated before on the purpose to represent different vehicle arrivals at depots, all the customers in the same cluster will have the same release date.

Appendix 3. Parameter tests on ϑ_{thr}

ϑ_{thr}	6				8				10				12				14				16			
	HPSO		HGA		HPSO		HGA		HPSO		HGA		HPSO		HGA		HPSO		HGA		HPSO		HGA	
Instance ID	Z	T(s)	Z	T(s)	Z	T(s)	Z	T(s)	Z	T(s)	Z	T(s)	Z	T(s)	Z	T(s)	Z	T(s)	Z	T(s)	Z	T(s)	Z	T(s)
E10-2-4-3-1	144.8	1.7	144.8	1.6	144.8	1.5	144.8	1.3	144.8	1.5	144.8	1.3	144.8	1.7	144.8	1.2	144.8	1.7	144.8	1.9	144.8	1.7	144.8	2.7
E10-2-4-3-2	189.3	2.9	189.3	3.6	189.3	2.4	189.3	2.8	189.3	2.2	189.3	2.4	189.3	2.4	189.3	2.6	189.3	2.6	189.3	2.7	189.3	2.8	189.3	3.6
E20-2-4-3-1	274.1	107	266.7	133	259.7	92	259.7	115	259.7	114	259.7	98	259.7	107	274.1	122	259.7	155	266.7	121	288.3	107	290.1	131
E20-2-4-3-2	294.9	109	307.2	88	294.9	106	294.9	96	294.9	98	294.9	102	294.9	107	294.9	109	294.9	102	294.9	97	315.2	125	307.2	114
E30-3-6-3-1	426.2	385	396.6	449	393.4	575	396.6	471	384.9	482	385.9	529	384.9	614	**358**	784	413.5	544	390.1	574	417.8	308	447.2	474
E30-3-6-3-2	544.5	427	560.3	396	571.8	534	**528.5**	552	547.9	674	543.9	663	552	628	575.4	597	610.7	578	595	613	595	615	610.7	577
E40-4-8-3-1	822.9	497	785.9	536	643.6	544	669.8	469	643.6	571	**586**	597	693.6	632	785.9	685	714.2	532	930	495	971	377	931	463
E40-4-8-3-2	1073.1	344	916.7	407	729.3	529	**675.4**	584	783.4	492	**675.4**	627	783.4	539	792.5	473	785.6	582	768.4	575	878	502	868	577
E60-6-12-3-1	1551	577	1619.1	659	1508.4	858	1564.8	1227	**1506.4**	1137	1516.8	1145	1599.3	842	1510.8	631	1606.9	728	1752.3	594	1637.6	475	1827.8	429
E60-6-12-3-2	2286.4	827	1758.5	615	2088.1	492	1758.5	544	1849.1	636	**1619.1**	589	1889.5	675	1903.6	430	2419.3	417	2064.2	528	2190.9	518	2419.3	476
E80-8-16-3-1	3692.3	509	3394.9	545	3623.3	614	3394.9	582	2708.1	743	2939.2	639	2708.1	1066	**2482**	1254	3466.7	367	2898.6	428	3623.3	396	2898.6	559
E80-8-16-3-2	7147.9	747	6823.4	683	5435.3	658	4198.5	539	4330.7	1130	4198.5	815	4981.8	586	6097.2	552	5435.3	588	**4187.5**	542	7147.9	429	6823.4	657

Appendix 4. Experiments on two variants

The proposed heuristics (HPSO and HGA) can also be applied in some variants of the Multi-D&T VRPTW-R, here we list two variants, multi-trip vehicle routing problem with time windows and release dates (MTVRPTWR) and multi-depot vehicle routing problem with time windows and release dates (MDVRPTWR). Table Ap.2 shows the comparison between the heuristics and CPLEX in small-scale instances for MTVRPTWR, Table Ap.3 shows the comparison for MDVRPTWR.

Table Ap.2: Comparison of the heuristics for MTVRPTWR

Instance ID	CPLEX		HPSO		Gap(%)	T-ratio	HGA		Gap(%)	T-ratio
	Z_c	$T_c(s)$	Z_p	$T_p(s)$	$(Z_p - Z_c)/Z_c$	T_p/T_c	Z_g	$T_g(s)$	$(Z_g - Z_c)/Z_c$	T_g/T_c
F5-1-1-3	58.2	0.6	58.2	0.7	0	1.17	58.2	1.2	0	2
F5-1-2-3	58.2	0.6	58.2	0.7	0	1.17	58.2	2.7	0	4.5
F5-1-3-3	58.2	0.5	58.2	0.9	0	1.8	58.2	3.5	0	7
F15-1-2-3	261.8	10.9	261.8	15	0	1.38	261.8	42	0	3.85
F15-1-3-3	219.2	7.6	219.2	8.6	0	1.13	219.2	19	0	2.5
F15-1-4-3	219.2	9.4	219.2	6.7	0	0.71	219.2	16	0	1.7
F25-1-3-3	427.5	857	427.5	67	0	0.08	427.5	219	0	0.26
F25-1-4-3	384.5	>3600	364	295	-5.33	/	386.8	71	0.6	/
F25-1-5-3	359.2	>3600	356.1	125	-0.86	/	357.9	147	-0.36	/)
Avg.		**126.66**		**57.7**	**-0.69**	**1.06**		**57.93**	**0.03**	**3.12**

Table Ap.3: Comparison of the heuristics for MDVRPTWR

Instance ID	CPLEX		HPSO		Gap(%)	T-ratio	HGA		Gap(%)	T-ratio
	Z_c	$T_c(s)$	Z_p	$T_p(s)$	$(Z_p - Z_c)/Z_c$	T_p/T_c	Z_g	$T_g(s)$	$(Z_g - Z_c)/Z_c$	T_g/T_c
F5-2-2-1	35.4	0.5	35.4	2.3	0	4.6	35.4	2.4	0	4.8
F5-2-4-1	35.4	0.5	35.4	2.7	0	5.4	35.4	2.5	0	5
F5-3-3-1	35.4	0.4	35.4	2.5	0	6.25	35.4	1.8	0	4.5
F15-2-4-1	190.3	2.3	190.3	13	0	5.65	190.3	8.8	0	3.83
F15-3-6-1	156.2	2.8	156.2	22	0	7.86	156.2	16	0	5.71
F15-4-8-1	156.2	4.1	156.2	24	0	5.85	156.2	17	0	4.15
F25-2-6-1	306.2	126	306.2	77	0	0.61	306.2	78	0	0.62
F25-3-6-1	297.9	97	301.6	54	1.24	0.56	297.9	52	0	0.54
F25-4-8-1	298.4	78	302.4	59	1.34	0.76	298.4	46	0	0.59
Avg.		**34.6**		**28.5**	**0.29**	**4.17**		**24.9**	**0**	**3.3**

Table Ap.2 and Table Ap.3 denote that the heuristics can be applied in MTVRPTWR and MDVRPTWR as the best solution can be obtained in most of the instances. The difference is that HPSO performs better in MTVRPTWR while it shows an opposite in MDVRPTWR.