CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Crowd-Shipping with Time Windows and Transshipment Nodes

Giusy Macrina
Luigi Di Puglia Pugliese
Francesca Guerriero
Gilbert Laporte

February 2019

CIRRELT-2019-05

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Crowd-Shipping with Time Windows and Transshipment Nodes

Giusy Macrina[1], Luigi Di Pugliese[1], Francesca Guerriero[1, *], Gilbert Laporte[2]

[1] Department of Mechanical, Energy and Management Engineering, University of Calabria, 87036, Rende (CS), Italy

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Management Sciences, HEC Montreal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

Abstract. Crowd-shipping is a delivery policy in which, in addition to standard vehicle routing practices, ordinary people accept to deviate from their route to deliver items to other people, for a small compensation. In this paper we consider a variant of the problem by taking into account the presence of intermediate depots in the service network. The occasional drivers can decide to serve some customers by picking up the parcels either from the central depot or from an intermediate one. The objective is to minimize the total cost, that is, the conventional vehicle cost, plus the occasional drivers' compensation. We formulate the problem and present a variable neighborhood search heuristic. To analyze the benefit of the crowd-shipping transportation system with intermediate depots and to assess the performance of our heuristic, we consider small and large-size instances generated from the Solomon benchmarks. A computational analysis is carried out with the aim of gaining insights into the behavior of both conventional vehicles and occasional drivers, and of analyzing the performance of our methodology in terms of effectiveness and efficiency. Our computational results show that the proposed heuristic is highly effective and can solve large-size instances within short computational times.

Keywords. Logistics, variable neighborhood search, on-line retailing, sharing economy, occasional drivers.

Acknowledgment. This work was supported by MIUR “PRIN 2015” funds, project: “Transportation and Logistics in the Era of Big Open Data” - 2015JJLC3E_003 - CUP H52F15000190001.

---

* Corresponding author: francesca.guerriero@unical.it

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2019

1 Introduction

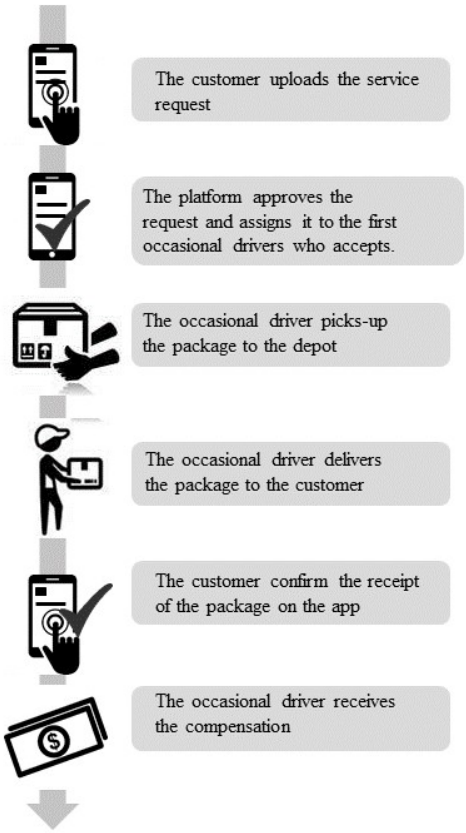
The fast and substantial growth of on-line retailers and the continuous search for new ways of speeding up deliveries and of overcoming the traditional problems of last-mile and same-day deliveries have generated some interest in crowd-shipping. This practice is related to the concept of "sharing economy", and allows some deliveries, that are usually performed by a company, to be outsourced to a large pool of individuals. It offers a new way of achieving transportation efficiency by exploiting underused assets. The reader is referred to [8] for a review of crowd-logistics, and to [19] for a complete survey of global trends in transportation and city logistics. The idea of crowd-shipping is as follows: an individual (whom we call occasional driver) who is travelling on his route, accepts to deviate from it to deliver items to other individuals, for a small compensation. In this context the exploited resources are commonly private cars which are often underused assets. The advantages of crowd-shipping are numerous and are not only related to economic issues:

- Costs saving: generally the compensation for the occasional drivers is less than the standard drivers' salary.
- No infrastructure: crowd-shipping does not require any significant infrastructure.
- Flexibility: while the traditional deliveries are fixed and planned in advance, for crowd-shipping fast delivery is the key factor.
- Reduced environmental impacts: sharing vehicles can lead to a reduction in polluting emissions, energy consumption, noise and traffic congestion.

Crowd-shipping requires a platform in order to connect occasional drivers with customers. When a customer buys some goods and requires a fast service delivery, he has to submit the on-line service request via a phone or a computer application. If he applies and accepts crowd-shipping delivery, the platform then sends the information needed to make the delivery. Once the delivery is completed, the occasional driver receives her compensation. Figure 1 summarizes the crowd-shipping process.

Several large on-line retailers, such as Walmart, DHL and Amazon, have started to implement and use platforms for crowd-shipping. In 2013, Walmart announced a plan to outsource some of its deliveries by asking in-store customers to deliver for a small compensation one or more orders placed by on-line customers (see Barr & Wohl [5]). In other words, if a shopper is in a Walmart store and has some free time, he/she may accept to deliver orders to other shoppers on the way to his/her destination. DHL experimented with crowd-shipping in Stockholm between September and December 2013 with a pilot last-mile service called MyWays (see Landa [13]). Using a specific smartphone application, the service connected individuals, mostly students, who asked for flexible deliveries with those offering to transport packages along their normal routes. The experiment was positive and many customers received or delivered packages by using MyWays. In June 2015, Amazon launched Amazon Flex (see Besinger [7]), its new crowd-shipping service that is still used in more than 30 cities in the world. To become an occasional driver for Amazon it is necessary to own an Amazon account, to satisfy some prerequisites, and to install the application. In particular, Amazon Flex offers several delivery opportunities based on the time window within which a package has to be delivered: three or more hours, one or two hours, less than one hour. An occasional driver may also choose from among two ways to pick

Figure 1: Crowd-shipping process



up delivery blocks: by choosing dates on a calendar, in which case the platform sends delivery offers for these dates, or by checking for available blocks on the Amazon Flex home screen.

Crowd-shipping is a recent topic and the relevant literature is therefore limited. Arslan et al. [4] reviewed and analyzed the potential benefits of crowd-shipping. They considered a peer-to-peer platform, taking into account the possibility of using traditional vehicles and ad hoc vehicles, and presented a rolling horizon framework and an exact algorithm to solve the route planning problem. Archetti et al. [3] introduced the vehicle routing problem with occasional drivers (VRPOD). In this work, the authors supposed that the company can make deliveries not only by using its own fleet of capacitated vehicles, but also by resorting to occasional drivers (ODs). These authors proposed an integer programming formulation for the VRPOD, and then developed a multi-start heuristic, which combines tabu search and variable neighborhood search (VNS). Starting from the VRPOD model of Archetti et al. [3], Macrina et al. [15] introduced three innovative aspects in the problem. They first considered time windows constraints for customers and ODs. Second, they allowed multiple deliveries for ODs. Third, they modelled the split and delivery policy for ODs. They tested and compared the proposed models with that of Archetti et al. [3] and showed the benefits of allowing multiple deliveries and of using the split and delivery policy. In both [3] and [15], the ODs must pick up the parcels at the depot and then deliver them to the customers.

Raviv and Tezer [18] studied a variant of the crowd-shipping problem in which the deliveries are carried out by external couriers. The delivery policy is based on a network of automatic service points used as a drop-off, pickup, and intermediate transfer points, where parcels can be dropped off by a courier and picked up again by another one. They developed a stochastic dynamic algorithm for a simplified version of the problem, where capacity constraints are not taken into account, as well as a greedy heuristic.

In this paper, as in Raviv and Tezer [18], we consider a scenario in which transshipment nodes are included in the service network. In particular, the parcels belonging to the transshipment nodes will be delivered by the ODs. The introduction of these nodes allows the transportation system to be more appealing to the ODs. Indeed, having transshipment nodes that act as intermediate depots closer to the delivery area could allow more ODs to be available to perform a task. In addition, the deviation of the ODs from their conventional route to the transshipment nodes could be less than the deviation with respect to the depot, which leads to a smaller compensation. However, the transshipment nodes have to be served by the classical vehicles in order to make the parcels available. Assuming the position of the transshipment nodes are known, one has to determine how many parcels have to be transferred from the depot to each transshipment node, at what time, and which parcel should be assigned to each transshipment node. All these decisions are related to the behaviour of the ODs. Indeed, each OD can choose to deliver parcels to some customers only if these parcels are available in a specific transshipment node and before a certain time. We allow only one pickup operation for each OD. This means that the parcels of the customer served by a OD must be available at the same transshipment node. We associate a time window with each customer and each OD. We consider the case in which a subset of customers must be served by the ODs. This restriction makes sense in the context of urban delivery parcels where the presence of classical vehicles is forbidden to ensure traffic reduction. However, the ODs can serve all customers, by possibly picking-up the parcels from the depot.

The use of transshipment nodes in the delivery process has been well studied. The transportation system is configured as a two-level system where the first level connects the central

depot with the transshipment nodes, and the second level connects the transshipment nodes with the customers. Two vehicle fleets are considered: one dedicated for the first level deliveries and the other to the second level. The scientific literature refers to this problem as a two-echelon VRP (2E-VRP). The 2E-VRP was introduced in [14], where a vehicle flow model was defined by considering several distribution modes for the whole system. Perboli et al. [16] proposed a network flow formulation along with valid inequalities. Later, in [17] a new formulation and a matheuristic procedure were presented. In [9] a general formulation was provided in the context of city logistics. For more details on the 2E-VRP, the reader is referred to [10] for contributions until 2015. Recently, a neighborhood search and set covering hybrid heuristic was proposed [1]. The 2E-VRP with multi-depots was addressed in [21]. Belgina et al. [6] considered the 2E-VRP with simultaneously pickup and delivery operations. They proposed a mathematical formulation and an algorithm based on variable neighborhood descent and local search.

Jie et al. [12] modeled and solved the 2E-VRP by considering electric vehicles for both first and second level, whereas in [2], the deliveries for the second level are performed by using bikes which, starting from a depot, pick up the parcel from the intermediate depots and deliver them to the customers. Gragier et al. [11] considered the 2E-VRP with multi-trips and solved it by means of an adaptive large neighborhood search metaheuristic. In [12, 2, 11], synchronization issues between the first and the second levels are taken into account since no storage is allowed at the intermediate depots.

Our problem differs from the classical 2E-VRP since classical vehicles supply not only the transshipment nodes, but can also serve customers. In our problem, the customers can be served either by classical vehicles or by ODs and this is a decision. Hence, we cannot know a priori which customer is served by classical vehicles. While both customers and transshipment nodes are delivery nodes for classical vehicles, for the ODs the customers are delivery nodes whereas transshipment nodes are used for pickups. In addition, we consider capacitated transshipment nodes, hence storage is allowed.

The contribution of this paper is twofold. We present a new variant of the crowd-shipping routing process where transshipment nodes are considered. We propose a mixed integer program (MIP) for our problem. We also develop a metaheuristic in order to efficiently solve the problem. We highlight that the proposed model is used as benchmark with the aim of evaluating the defined metaheuristic in terms of both effectiveness and efficiency.

The remainder of the paper is organized as follows. In Section 2 we model the VRPOD with time windows and transshipment nodes (VRPODTN). In Section 3 we describe the proposed metaheuristic for the VRPODTN. In Section 4 we perform computational experiments and we present the results. Section 5 summarizes our conclusions.

2 The vehicle routing problem with occasional drivers, time windows and transshipment nodes

We use the following notation in our model (see Table 1 for a summary). Let C be the set of customers that can be served by either the classical vehicles or by the ODs, let s be the origin node and let t be the destination node for the classical vehicles, i.e. those belonging to the company. Let R be the set of customers that must be served by the ODs. Let K be the set of available ODs, U and V be the sets storing the origin and the destination nodes $u_k, v_k, \forall k \in K$, respectively. We assume that R and K are disjoint sets. Let T be the set of transshipment

nodes. We define the node set as $N = C \cup R \cup T \cup U \cup V \cup \{s, t\}$.

A cost c_{ij} and a travel time t_{ij} are associated with each node pair $i, j \in N$. Note that both c_{ij} and t_{ij} satisfy the triangle inequality. Each node $i \in C \cup R$ has a time window $[e_i, l_i]$, a demand d_i, and a service time t_i. Let t_{u_k} and l_{v_k} be the time at which OD k is available to perform a delivery, and the time at which she must reach her own destination v_k, respectively. P is the number of available classical vehicles, Q is the capacity of the classical vehicles, Q_p^T is the capacity of the transshipment node p, and Q_k^K is the capacity of OD k. The classical vehicles start their route from node s, serve possibly a subset of nodes in $C \cup T$ and reach the destination node t. The demand associated with each transshipment node p is a variable, defined as δ_p, whose value depends on the behaviour of the ODs.

Let x_{ij} be a binary variable equal to 1 if and only if a classical vehicle traverses arc (i, j). Let y_{ij} be the parcel flow passing through arc (i, j). Let s_i be the arrival of a classical vehicle at customer i. Let z_i be a binary variables taking value 1 if node $i \in C \cup T$ is served by the classical vehicles, and 0 otherwise.

To simplify the model, we define D as the number of parcels delivered by the classical vehicles. We have to distinguish between service to the customers and service to the transshipment nodes. Indeed, in the latter case, we have a non-linear component. The definition of D is

$$D = \sum_{i \in C} z_i d_i + \sum_{p \in T} z_p \delta_p. \tag{1}$$

To linearize the term $z_p \delta_p$, we introduce a new variable $\bar{\delta}_p, \forall p \in T$. In particular, $\bar{\delta}_p = z_p \delta_p$ and the following constraints are included in the model:

$$\bar{\delta}_p \leq \delta_p \; \forall p \in T \tag{2}$$
$$\bar{\delta}_p \leq M z_p \; \forall p \in T \tag{3}$$
$$\bar{\delta}_p \geq \delta_p + M(z_p - 1) \; \forall p \in T, \tag{4}$$

where M can be set to $\sum_{i \in C \cup R} d_i$. The constraints that model the behaviour of the classical vehicles are

$$\sum_{j \in C \cup T} x_{sj} \leq P \tag{5}$$

$$\sum_{j \in C \cup T} x_{sj} - \sum_{j \in C \cup T} x_{jt} = 0 \tag{6}$$

$$\sum_{j \in C \cup T} x_{ij} - \sum_{j \in C \cup T} x_{ji} = 0, \ \forall i \in C \cup T \tag{7}$$

$$\sum_{j \in C \cup T \cup \{s\}} y_{ji} - \sum_{j \in C \cup T \cup \{t\}} y_{ij} = \begin{cases} -D & \text{if } i = s \\ 0 & \text{if } i = t \\ z_i d_i & \text{if } i \in C \\ \bar{\delta}_i & \text{if } i \in T \end{cases} \forall i \in C \cup T \cup \{s,t\} \tag{8}$$

$$y_{ij} \leq Q x_{ij}, \ \forall i,j \in C \cup T \cup \{s,t\}, i \neq j \tag{9}$$

$$s_j \geq s_i + t_i + t_{ij} - M(1 - x_{ij}), \ \forall i,j \in C \cup T \cup \{s,t\}, i \neq j \tag{10}$$

$$z_i \geq \sum_{j \in C \cup T \cup \{t\}} x_{ij}, \ \forall i \in C \cup T \tag{11}$$

$$e_i - M\left(1 - z_i\right) \leq s_i \leq l_i + M\left(1 - z_i\right), \ \forall i \in C. \tag{12}$$

Constraint (5) guarantees that at most P vehicles leave the depot. Constraints (6) ensure that each vehicle leaving the depot ends its route at node t. Constraints (7) balance the vehicles entering and leaving node i. Constraints (8) impose that the flow balance at each node is equal to the associated demand, except for the depot, where the exit flow is equal to the number of parcels that have to be delivered. Constraints (9) guarantee that the number of delivery parcels does not exceed the capacity of the vehicles. Constraints (10) define the arrival time at each node i. Constraints (11) define the variables z_i. Constraints (12) guarantee that each customer $i \in C$ is served within its time window. We note that the customers $i \in C$ can be served by either a classical vehicle or by an OD.

Each OD k, starting from her initial position u_k, deviates from her route to pickup, either at a transshipment node or at the depot, the parcels of the customers she serves. Let t_k be the time at which OD k becomes available.

Let $r^k_{pi}, \forall p \in T \cup \{s\}, i \in C \cup R, k \in K$ be binary variables stating whether OD k serves customer i and picks up the parcels to deliver at transshipment node p. Let $f^k_p, \forall p \in T \cup \{s\}, k \in K$ be continuous variables indicating the arrival time of OD k at transshipment node p. Moreover, let $\tau_i, \forall i \in C \cup R$ be the arrival time of some OD at customer i.

The operations of the ODs are modelled by the following set of constraints:

$$\sum_{p\in T\cup\{s\}}\sum_{i\in C\cup R} r_{pi}^k \leq 1,\ \forall k \in K \tag{13}$$

$$\sum_{p\in T\cup\{s\}}\sum_{k\in K} r_{pi}^k = 1,\ \forall i \in R \tag{14}$$

$$f_p^k \geq t_{u_k} + t_{u_k p} - M\left(1 - \sum_{i\in C\cup R} r_{pi}^k\right),\ \forall p \in T\cup\{s\}, k \in K \tag{15}$$

$$\tau_i \geq f_p^k + t_p + t_{pi} - M(1 - r_{pi}^k),\ \forall i \in C\cup R, p \in T\cup\{s\}, k \in K \tag{16}$$

$$e_i \leq \tau_i \leq l_i,\ \forall i \in R \tag{17}$$

$$e_i - M\left(1 - \sum_{p\in T\cup\{s\}}\sum_{k\in K} r_{pi}^k\right) \leq \tau_i \leq l_i + M\left(1 - \sum_{p\in T\cup\{s\}}\sum_{k\in K} r_{pi}^k\right),\ \forall i \in C \tag{18}$$

$$\tau_i + t_i + t_{iv_k} \leq l_{v_k} + M(1 - r_{pi}^k),\ \forall i \in C\cup R, p \in T\cup\{s\}, k \in K \tag{19}$$

$$r_{pi}^k \leq \frac{Q_k^K}{d_i},\ \forall i \in C\cup R, p \in T\cup\{s\}, k \in K \tag{20}$$

$$\delta_p = \sum_{i\in C\cup R}\sum_{k\in K} d_i r_{pi}^k,\ \forall p \in T \tag{21}$$

$$z_p \geq \frac{\delta_p}{\sum_{i\in C\cup R} d_i},\ \forall p \in T. \tag{22}$$

Constraints (13) impose that each OD k can serve at most one customer and pick up the parcels at one transshipment node. Constraints (14) state that each customer $i \in R$ is served by an OD. Constraints (15) define the time at which OD k visits transshipment node p. Constraints (16) define the arrival time of OD k at node i when passing through transshipment node p. Constraints (17) and (18) define the time window constraints associated with each customer $i \in R$ and $i \in C$, respectively. We note that customer $i \in C$ can be served either by a classical vehicle or by an OD. Constraints (19) limit the route duration of OD k. Constraints (20) prevent the assignment of occasional driver k to customer i if the capacity Q_k^K of the occasional driver is not sufficient to deliver the request d_i of customer i. Constraints (21) define the number of parcels that must be available at transshipment node p in order to guarantee the delivery process performed by the ODs. In other words, these constraints define the number of parcels that the classical vehicles have to deliver to the transshipment nodes $p \in T$. Constraints (22) impose that transshipment node $p \in T$ must be served if it is used by the ODs to deliver parcels.

We consider capacitated transshipment nodes. Hence, each transshipment node $p \in T$ is allowed to store at most Q_p^T parcels. The following constraints are added to the model:

$$\delta_p \leq Q_p^T,\ \forall p \in T. \tag{23}$$

Of course, it is necessary to synchronize the classical vehicles and the ODs with respect to the delivery and pickup operations at the transshipment nodes p. To this end, the following constraints are introduced:

$$s_p \leq f_p^k + M\left(1 - \sum_{i\in C\cup R} r_{pi}^k\right),\ \forall p \in T, k \in K. \tag{24}$$

Constraints (24) impose that OD k can pickup parcels from node p only after node p has been served by a classical vehicle. In addition, the customer belonging to the set C must be served either by a classical vehicle or by an OD. The following constraints are defined:

$$z_i + \sum_{p \in T \cup \{s\}} \sum_{k \in K} r_{pi}^k = 1, \ \forall i \in C. \tag{25}$$

The objective function is the minimization of the overall cost. In particular, we aim at minimizing the routing cost of the classical vehicle given by

$$Z_{cv} = \sum_{i \in C \cup T \cup \{s\}} \sum_{j \in C \cup T \cup \{t\}} c_{ij} x_{ij}$$

and the compensation to the ODs. In particular, we compute the cost incurred by the ODs who must deviate from their normal route, whose cost is $c_{u_k v_k}$, to performs deliveries. The compensation cost for OD $k \in K$ is given by

$$Z_k = \rho \left(\sum_{p \in T \cup \{s\}} \sum_{i \in C \cup R} \left(c_{u_k p} + c_{pi} + c_{i v_k} - c_{u_k v_k} \right) r_{pi}^k \right).$$

The objective function of our model is defined as

$$\text{minimize} \ \ Z_{cv} + \sum_{k \in K} Z_k. \tag{26}$$

3 Variable neighborhood search

This section details our VNS for the VRPODTN. Algorithm 1 presents the VNS scheme. First, we generate an initial solution ϕ, then we apply a *Shaking* phase to perturb ϕ in order to explore the neighborhoods and the Variable Neighborhood Descent (VND) to improve the solution. In what follows, VND (ϕ') refers to VND applied to solution ϕ'.

Initial solution The initialization procedure is a two-phase matheuristic. We first find the best assignment of customers and transshipment nodes to the ODs. Thus, we formulate and then solve with CPLEX the following assignment model:

$$\text{minimize} \ \ \sum_{k \in K} Z_k + \sum_{i \in C} 2 c_{si} z_i + \sum_{p \in T} 2 c_{sp} z_p \tag{27}$$

subject to

(13)–(25)

$$\sum_{p \in T} \delta_p \leq PQ \tag{28}$$

$$\delta_p \leq Q, \ \forall p \in T. \tag{29}$$

Table 1: Sets, parameters and decision variables of the VRPODTS model

s	origin node for classical vehicles
t	destination node for classical vehicles
C	set of customers served by either classical vehicles or ODs
R	set of customers that must be served by the occasional drivers
T	set of transshipment nodes
K	set of available occasional drivers
U	set of u_k origins for the occasional drivers
V	set of v_k destinations for the occasional drivers
Parameters	
t_{u_k}	instant time in which the OD K is available to perform a delivery starting from the own origin u_k
l_{v_k}	instant time in which the OD K must reach the own destination v_k
c_{ij}	travel cost from node i to node j
t_{ij}	travel time from node i to node j
$[e_i, l_i]$	time windows of node i
d_i	demand of customer i
t_i	service time of node i
P	number of classical vehicles
Q	capacity of classical vehicles
Q_k^k	capacity of occasional driver k
Q_p^T	capacity of transshipment node p
Variables	
x_{ij}	binary decision variable indicating if arc (i, j) is traversed by a classical vehicle
y_{ij}	decision variable specifying the parcel flow on arc (i, j) associated with a classical vehicle
s_i	decision variable specifying the arrival time of the classical vehicle to customer $i \in C$
r_{pi}^k	binary decision variable indicating whether OD k serves customer $i \in C \cup R$, picking up parcels at transshipment node p
f_p^k	decision variable specifying the arrival time of the occasional driver k at transshipment node p
τ_i	decision variable specifying the arrival time of the occasional driver k at customer $i \in C \cup R$
δ_p	decision variable specifying the quantity of parcel available at transshipment node $p \in T$
z_i	binary decision variable specifying whether node $i \in C \cup T$ is served by the classical vehicles

Algorithm 1 Variable neighborhood search

Input set of neighbourhood N^h, for $h = 0, ..., h_{max}$
Initialization Initial solution ϕ
while $h \leq h_{\max}$ and $k \leq k_{\max}$ **do**
  $\phi' \leftarrow$ **Shaking** (ϕ)
  $\phi'' \leftarrow$ **VND** (ϕ')
  if $f(\phi'') < f(\phi)$ **then**
    $\phi \leftarrow \phi''$;
    $h \leftarrow 0$
  else
    $h \leftarrow h + 1$
  end if
  $k \leftarrow k + 1$
end while
return ϕ

where the objective function (27) takes into account the cost of the ODs, i.e., Z_k, and the routing cost of the classical vehicles. We highlight that the term Z_k is not a constant since it depends on variables r^k_{pi} included in constraints (13)–(25).The model (27)–(29) does not contain x variables, thus we approximate the routing cost. In particular, we consider the cost of a simple route that starts at node s, serves either node $i \in C$ not served by the ODs, i.e., c_{si} or the transshipment nodes used by the ODs, i.e., c_{sp} and ends at t.

Constraints (28) and (29) take into account the capacity constraint for the classical vehicles. It is important to point out that both constraints (28) and (29) are needed. On the one hand, if we consider only constraint (28), it is possible to assign to transshipment node p a demand δ_p that cannot be deliver by the classical vehicle due to capacity limitation. We do not allow split policy on transshipment nodes. On the other hand, if we consider only constraints (29), it is possible that the total demand of the transshipment nodes cannot be delivered by the fleet of vehicles considered. This can happen since we do not assume any limitation on the number of transshipment nodes and classical vehicles and we do not consider multi-trip.

Once the assignment problem has been solved, we can have a set of unserved customers $C' \subseteq C$ and a set of activated transshipment nodes $T' \subseteq T$. We then apply an insertion heuristic adapted for the VRPODTN, which takes into account the heterogeneity of the fleet. Since the nodes in T' have to be served by classical vehicles, we first try to serve both C' and T' with these vehicles. The initial tour is composed of the origin and destination nodes. The heuristic inserts a new node in the tour in the best feasible position, i.e., where it causes the least increase in the tour cost. We first consider the customers in T', and then we serve the customers in C'. Since the initial solution may be infeasible, we apply a repair phase. In particular, if some customers in C' are still not served by a classical driver, we try to assign them to the available ODs, in the best feasible position, imposing that the OD must take the parcel to the depot. If no ODs are available or it is not possible to serve some customers in C', or if some transshipment nodes are not served, we apply the local search moves defined for the VND, and we then insert the unserved customers and transshipment nodes until a feasible solution has been generated.

Local search operators In order to generate the neighborhoods we use four different local search (LS) moves:

1. Move Node: This operator removes one node i from a route r and inserts it in another route r' in the best feasible position. We implemented four variants: classical to classical, classical to OD, OD to OD, OD to classical.

2. Swap Inter-Route: This operator removes one node i from a route r and one node j from another route r', $r \neq r'$, and inserts i into r' and j into r in the first feasible positions. We implemented four variants: classical to classical, classical to OD, OD to OD, OD to classical.

3. Swap Intra-Route: Given a classical route r, this operator changes the position of two nodes i and j while satisfying the constraints.

4. New Route: This operator initializes a new route r'. It removes one node i from a route r and inserts i in r'. We implemented two variants: classical and OD.

Considering all the aforementioned possible moves, we develop 11 LS operators.

Shaking The main goal of the shaking phase is to perturb the current solution. Thus, we randomly select and apply two different LS operators are ordered in non-decreasing score values, then we randomly choose two of them among the 11 available and we allow the current solution to worsen. To improve the shaking phase, we introduce a semi-random choice, namely after the first iteration we assign a score to each LS move. At the end of each VNS iteration, if there is an improvement in the solution cost, we increment the scores of the shaking moves; otherwise the scores are reduced. The LS operators are ordered in non-decreasing score values, then we randomly choose two of them among the first five.

Variable neighborhood descent Generally, in the VNS scheme the shaking process is followed by an LS phase, in which the h_{max} different LS operators are applied with the purpose of improving the current solution. We propose a VND heuristic described in Algorithm 2.

Algorithm 2 Variable neighborhood descent

Input the set of neighborhood N^h, for $h = 0, ..., h_{max}$
Initialization initial solution ϕ
$improved \leftarrow 0$, $k \leftarrow 0$
while $improved = false$ and $k \leq k_{\max}$ **do**
  $h \leftarrow 0$
  while $h \leq h_{\max}$ **do**
    $\phi' \leftarrow N^h(\phi)$
    if $f(\phi') < f(\phi)$ **then**
      $\phi \leftarrow \phi'$
      $improved \leftarrow 1$
    else
      $h \leftarrow h + 1$
    end if
  end while
  $k \leftarrow k + 1$
end while
return ϕ

4 Computational results

We now analyze the behaviour of the proposed model and that of the metaheuristic. The model and the metaheuristic are implemented in Java. The model is solved by CPLEX 12.7 and we impose a time limit of 900 seconds for the solution of an instance.

In Section 4.1 we illustrate the procedure we have developed to generate the instances. In Section 4.2, we summarize the main results highlighting the findings by analyzing the behavior of both the proposed model and metaheuristic. The details and an exhaustive explanation of the behavior are reported in Sections 4.3 and 4.4. In Section 4.3 we evaluate the performance of the model on small-size instances. In particular, we compare the solution with that obtained by a modified version of the model in which no transshipment nodes are considered. We then perform a sensitivity analysis by varying several parameters with the aim of studying the impact of both

the ODs and the classical vehicles. In Section 4.4 we analyze the performance of the proposed metaheuristic on a large set of instances. The tests are carried out by considering an Intel Core i7-4720HQ CPU, 2.60 GHz, 8GB RAM, machine under Windows 10 operating system.

4.1 Generation of the instances

We generated instances of the VRPODTN based on the Solomon VRPTW instances composed of 400 nodes [20]. In particular, we consider the VRPTW instances of classes C1, R1, and RC1.

Starting from the VRPTW benchmark instances we generate instances of the VRPODTN by considering several values for the number of customers that must be served by the ODs, i.e. $|R|$, those that can be served either by the classical vehicles or by the ODs, i.e. $|C|$, the number of transshipment nodes $|T|$, and the number of the ODs $|K|$. The VRPODTN instances are generated by applying the following steps.

- We divided the set of customers $\bar{C}$ of the VRPTW benchmark instances into three subsets, i.e., C_M, R_M, and T_M. Recalling that the VRPTW benchmark instances are composed of customers located at coordinate $(x_i, y_i), \forall i \in \bar{C}$ of a rectangular field characterized by a maximum and minimum values for the X and Y axes, i.e., X_m, X_M, Y_m, and Y_M, let p_1 and p_2, $p_1 > p_2$, be two numbers belonging to the range $[0, 1]$, (see Figure 2). Then
  - $C_M = \{i \in \bar{C} : y_i > (y_s - Y_m)p_1\}$,
  - $R_M = \{i \in \bar{C} : y_i \leq (y_s - Y_m)p_2\}$,
  - $T_M = \{i \in \bar{C} : (y_s - Y_m)p_2 < y_i \leq (y_s - Y_m)p_1\}$.

  The rectangular field is divided into three subregions, SR_1, SR_2, and SR_3, based on the position of the depot s. The customers served by the ODs, i.e., those belonging set R, are located in SR_3, that is, they are far away from the depot s. The transshipment nodes are located between the depot and the customers $i \in R$, i.e., in SR_2. The customers $i \in C$ are located in the remaining area, i.e., SR_1. Figure 2 shows the partition of the field into the three subregions. We consider $p_1 = 0.9$ and $p_2 = 0.6$ for the generation of the instances.

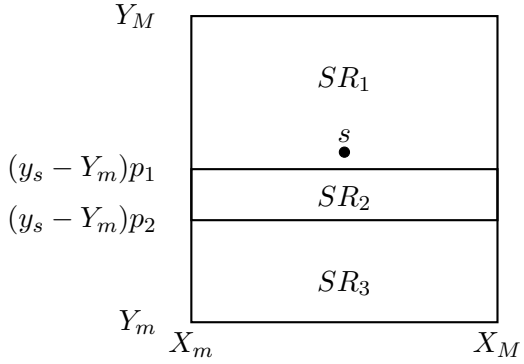


Figure 2: Partition of the field into the three sub-regions SR_1, SR_2, and SR_3 with input the position of the depot s and the parameters p_1 and p_2.

- We compute the set C and R, considering the first $|C|$ customers contained in C_M and the first $|R|$ customers contained in R_M, respectively.

- For the choice of the transshipment nodes, we include into subset T the nodes belonging to T_M, guaranteeing that they are spatially dispersed. In particular, given a distance threshold $\bar{d}$, a node $i \in T_M$, not yet included in T, is added to T only if the distance between i and the last inserted node into T is greater than $\bar{d}$. In the case $|T|$ is less than the chosen value, the insertion procedure is repeated by considering a lower value of $\bar{d}$.
- The set K is composed of the nodes $i \in R_M \cup C_M : i \notin R, i \notin C$. It follows that OD k coincides with customer i.
- The node u_k of the OD k coincides with the location of customer i, whereas, node v_k is chosen randomly in a circle of radius 10 with center a location of some customer $i \in R \cup C$.
- We set the number of available classical vehicles P equal to 3. We assume that the vehicles are identical and all have the same capacity Q.
- We slightly modified the time windows $[\bar{e}_i, \bar{l}_i]$ of the VRPTW benchmark instances. In particular, we set $l_i = \bar{l}_i + \gamma, \forall i \in C \cup R$, with γ randomly chosen in the interval $[50, 400]$.
- The time at which the OD becomes available for a delivery is computed as $t_k = \min\{\bar{e}_{\hat{i}}, \mu\}$, with μ randomly chosen in the interval $[0, e_m]$, where $e_m = \min_{i \in R}\{e_i\}$. The due date of each OD k is computed as $l_k = \max\{\bar{l}_{\hat{i}}, \epsilon\}$, with ϵ randomly chosen in the interval $[l_M, l_M + 200]$, where $l_M = \max_{i \in R}\{l_i\}$. The capacity Q_k^K is set equal to $\max_{i \in R}\{d_i\} + 10$. This choice guarantees that all customers $i \in R$ can be served by each OD.
- We generate several instances by considering different values for $|C|$, $|R|$, $|T|$ and $|K|$, where $|C| \in \{5, 10, 15, 30, 40, 50\}$, $|R| \in \{10, 20, 30, 40, 50\}$, $|T| \in \{2, 4, 6\}$, and $|K| = |C| + |R|$.
- We consider four different values for the vehicle capacity, i.e., $Q = \alpha \frac{\sum_{i \in C \cup R} d_i}{P}$, with $\alpha \in \{0.70, 1, 1.30, 2\}$.
- We consider three values for the capacity Q_p^T of each transshipment node $p \in T$, i.e., $Q_p^T = \beta \frac{\sum_{i \in C \cup R} d_I}{|T|} + 1$ with $\beta \in \{1, 1.3, 2\}$.
- We consider three values for the compensation of the ODs, i.e., $\rho \in \{0.75, 1.5, 2.25\}$.

Since, we modified the time windows of the VRPTW benchmark instances, we restrict ourselves to the first VRPTW benchmark instance of each class.

4.2 Main findings

We analyze the effectiveness of using transshipment nodes in the transportation system with ODs. For this purpose, we solve the proposed model by letting $|T| = 0$, i.e., we solve a VRPOD. We also conduct a sensitivity analysis on the parameters of the model, i.e., the number of transshipment nodes ($|T|$), the vehicle capacity (α), the transshipment node capacity (β), and the compensation of the ODs (ρ). We can summarize the results obtained as follows:

- VRPODTN allows a cost saving of about 31% compared with VRPOD;
- the total cost decreases when the number of available transshipment nodes, the vehicle capacity, and the transshipment node capacity increase;

- for high values of the compensation ρ, the number of ODs deliveries decreases and the total cost increases.

For more details on model evaluation, the reader is referred to Section 4.3. In Section 4.3.1 we describe the results of the comparison between VRPODTN and VRPOD. Section 4.3.2 reports the sensitivity analysis.

We tested the proposed metaheuristic with the aim of evaluating its performance both in terms of solution quality and of efficiency. We can draw the following considerations from the computational results:

- the VNS solves the considered instances with an optimality gap of about 3%, on average;
- the computational overhead is very limited compared with the time required by CPLEX, and the VNS is about 33 times faster than CPLEX, on average;
- for the instances not solved to optimality by CPLEX, the VNS obtains, within the same computational time, solutions with a cost 80% lower than that obtained by CPLEX, on average.

A detailed description of the computational results with an in-depth analysis is carried out in Section 4.4.

4.3 Model evaluation

In this section we evaluate our model on the instances described in Section 4.1. We restrict our attention to the instances with $|C| \in \{5, 10, 15\}$. For each instance class and value of $|C|$, we consider $5 \times 3 \times 4 \times 3 \times 3$ VRPODTN instances, where the factors represent the different values for $|R|$, $|T|$, α, β, and ρ, respectively. We refer to these instances as small-size instances.

4.3.1 Comparison between VRPODTN and VRPOD

We analyzed the two problems, i.e., the VRPODTN and VRPOD, in terms of cost. Table 2 shows the average results obtained by varying the number of transshipment nodes ($|T|$) for the three classes of instances. In particular, we report the total cost (26) under column Obj, the percentage decreasing of the Obj related to VRPODTN with respect to VRPOD under column %d, the routing cost of the classical vehicles under column Z_{cv}, and the cost associated with the deliveries performed by the ODs under column Z_k. The column #vs reports the average number of classical vehicles used. The superscripts in the column Obj represent the number of instances for which the execution time of CPLEX exceeds the time limit. We only have a feasible solution for these instances. The aim of this section is to show the benefit of including transshipment nodes in terms of cost. Overall, we observe a cost reduction when the transshipment nodes are considered. In addition, since all VRPOD instances are solved to optimality but VRPODTN instances are not, the observed cost reduction is underestimated.

		R1					C1					RC1				
$\|C\|$	$\|T\|$	Obj	%d	Z_{cv}	Z_k	#vs	Obj	%d	Z_{cv}	Z_k	#vs	Obj	%d	Z_{cv}	Z_k	#vs
5	0	4989.19		189.62	4799.57	0.95	5657.34		89.85	5567.48	0.75	5036.22		50.14	4986.07	0.33
	2	3132.69	37%	394.64	2738.05	2.20	4654.52	18%	332.56	4321.96	1.61	4200.66	17%	313.08	3887.59	1.85
	4	2772.72	44%	424.16	2348.56	2.52	3806.78	33%	354.24	3452.54	2.20	3883.47	23%	338.91	3544.56	2.57
	6	2790.69	44%	420.13	2370.57	2.69	3933.26	30%	356.02	3577.24	2.08	3704.80	26%	354.38	3350.42	2.43
10	0	5013.14		290.56	4722.57	1.17	5677.87		203.13	5474.74	1.47	5139.87		210.63	4929.23	0.87
	2	3165.03	37%	469.94	2695.08	2.18	4653.52	18%	390.82	4262.69	1.80	4214.44	18%	368.03	3846.41	1.90
	4	2804.01	44%	451.35	2352.66	2.49	3751.53	34%	418.64	3332.89	2.33	3879.16	25%	418.47	3460.69	2.66
	6	2802.80	44%	454.41	2348.39	2.64	3881.63^{10}	32%	415.11	3466.52	2.17	3695.57	28%	429.41	3266.16	2.50
15	0	5095.53		444.25	4651.28	1.73	5719.43		311.86	5407.57	1.54	5125.95		359.18	4766.78	1.47
	2	3208.62	37%	578.47	2630.15	2.25	4686.17	18%	455.67	4230.50	1.92	4194.72	18%	428.03	3766.69	2.07
	4	2852.40	44%	524.79	2327.62	2.49	3714.13^{5}	35%	467.92	3246.21	2.37	3831.67	25%	458.83	3372.85	2.69
	6	2851.12^{2}	44%	521.41	2329.70	2.60	3848.58^{67}	33%	478.91	3369.67	2.34	3628.95^{4}	29%	468.75	3160.20	2.62
Avg	0	5032.62		308.14	4724.48	1.28	5684.88		201.61	5483.26	1.25	5100.68		206.65	4894.03	0.89
	2	3168.78	37%	481.01	2687.76	2.21	4664.74	18%	393.02	4271.72	1.77	4203.27	18%	369.71	3833.56	1.94
	4	2809.71	44%	466.77	2342.94	2.50	3757.48^{5}	34%	413.60	3343.88	2.30	3864.77	24%	405.40	3459.37	2.64
	6	2814.87^{2}	44%	465.32	2349.55	2.64	3887.82^{77}	32%	416.68	3471.14	2.20	3676.44^{4}	28%	417.51	3258.93	2.52

Table 2: Analysis of the VRPODTN agains the VRPOD.

Focusing our attention on the class R1, the average numerical results highlight that the VRPODTN provides a solution with a total cost (see column %d) 37%, 44%, and 44% lower than the total cost of VRPOD, for $|T|$ equal to 2, 4, and 6, respectively. We observe the same trend for the class C1. However, in this case, the decrease in cost is less impressive. Indeed, Obj for the VRPODTN is 18%, 34%, 32% lower than the cost of VRPOD. We obtain similar results for the class RC1 where the total cost for the VRPODTN is 18%, 24%, and 28% lower than the cost of the VRPOD for $|T|$ equal to 2, 4, and 6, respectively.

Table 2 shows that the number of classical vehicles used in the VRPOD is lower than the number of vehicles used in the VRPODTN. This is to be expected, since the vehicles do not have to serve the transshipment nodes in the VRPOD. This behaviour explains the lower routing cost Z_{cv} in the VRPOD. However, the reduction in the routing cost Z_{cv} does not compensate for the higher cost Z_k incurred in the VRPOD.

4.3.2 Sensitivity analysis on model parameters

The behaviour of the proposed model on the small-size instances is analyzed by considering Obj, Z_{cv}, Z_k, #vs and the following statistics:

Time : computational time in seconds;

Ts : number of transshipment nodes used;

#ODs : number of deliveries performed by the ODs;

%ODsD : percentage of #ODs that pickup the parcels directly from the depot s;

%vsd : percentage of customers $i \in C$ that are served by the classical vehicles.

Tables 3–5 report the average numerical results for each classes, i.e., R1, C1, and RC1. The superscripts in the column Time represent the number of instances that are not solved to optimality within the time limit imposed. Tables 3–5 exhibit a similar performance, meaning that the model behaves quite similarly for different classes. From the average results collected in Tables 3–5, we can make following analyses.

Analysis on the number of available transshipment nodes ($|T|$). When varying the number of available transshipment nodes $|T|$, the total transportation cost varies as well. In particular, the higher $|T|$, the lower the value of Obj. This is expected, since a higher number of available transshipment nodes means a better organization of the deliveries for the ODs. Indeed, the values of %ODsD decreases when $|T|$ increases. This behaviour is more evident for the instances of class R1 (see Table 3). We observe that, on average, the cost Z_{cv} increases when $|T|$ increases. This is due to the larger number of vehicles used (see column #vs). For classes R1 and C1, we observe a slightly increase of the cost Obj from $|T| = 4$ to $|T| = 6$. This is because the lower cost Z_k does not compensate for the higher cost Z_{cv}.

Analysis on classical vehicles' capacity (α). As expected, the capacity of the classical vehicles influences the behaviour of both the classical vehicles and the ODs. As a consequence, a better value of Obj is observed for higher values of α, due to the decrease of both the values of Z_{cv} and Z_k. The lower routing cost Z_{cv} is expected since a lower number of vehicles are used

(see column #vs). The interesting result is that we also observe a reduction of Z_k. Since when α increases the number of transshipment nodes used increases, fewer ODs pick up the parcels from the depot (see column %ODsD). As a result, we have a better organization of the ODs.

Analysis on transshipment nodes' capacity (β). When varying the capacity of the transshipment nodes, the total cost varies as well. In particular, the higher β, the lower Obj. The reduction of Z_{cv} is justified by the reduction of Ts. Thus, fewer transshipment nodes have to be served. A higher capacity for the transshipment nodes allows a better assignment of the ODs' deliveries. Since picking up the parcels from the transshipment nodes is more convenient for the ODs than visiting the depot, we observe a reduction in the cost Z_k.

Analysis on the cost for ODs' deliveries (ρ). As expected, the higher the compensation for the ODs, the higher the Obj value. We observe an increase of both Z_k and Z_{cv}. We note that the ODs must serve all customers $i \in R$, but they can also serve some customers $i \in C$. Thus, on the one hand, when ρ increases the number of OD deliveries decreases, because these drivers serve fewer customers $i \in C$. A lower value of #ODs does not compensate for the higher compensation for each OD. Thus, a higher value of Z_k is observed. On the other hand, the number of the customers served by the classical vehicles increases. As a consequence, we observe an increase in Z_{cv}.

4.4 Heuristic evaluation

In this section, we analyze the behavior of the VNS. We first compare the solutions obtained with the heuristic with those obtained by CPLEX on small-size instances. We then test the heuristic procedure on large-size instances characterized by $|C| \in \{30, 40, 50\}$, $|R| \in \{10, 20, 30, 40, 50\}$ and we use the following configurations for the other parameters: $|T| = 6$, $\alpha = 2.0$, $\beta = 1.3$ and $\rho = 1.50$.

The rationale for these choices can be explained by considering the results of the sensitivity analyses carried out in Section 4.3.2. More specifically, the large-size instances are characterized by a larger number of available transshipment nodes ($|T| = 6$). In addition, from the results shown in Section 4.3.2, we observe that the larger the vehicle capacity, the better the organization of the delivery process of the ODs. This observation motivates the choice of $\alpha = 2$. The same considerations are valid for the parameter β. In this case, we choose an intermediate value for β since it guarantees a good compromise between the routing cost and the cost related to the OD deliveries. In addition, a high value for β implies a reduction in the number of transshipment nodes used. For the compensation value ρ of the ODs we use an intermediate value since the higher ρ, the lower the number of deliveries performed by the ODs.

Results on small-size instances We now present the results obtained on small-size instances. We set $k_{\max} = 10$ and $improve = 15$. Table 6 summarizes the average results obtained for small-size instances, grouped by the number of customers in C and by class. The first column displays the number of customers $|C|$ and the second one the class. The third column shows the average speedup, calculated as the ratio between the computational time required by CPLEX and the computational time taken by the heuristic. In the third column we report the percentage optimality gap on cost, calculated as Gap_c= $(\text{Objective}_\text{VNS} - \text{Objective}_\text{CPLEX})/\text{Objective}_\text{CPLEX}$.

The VNS is much less time consuming than CPLEX. In particular, the larger the size of C the larger is the value of speedup.

The VNS is also effective. Indeed the average gap is about 3%. In particular, Gap_c is about 2%, 3% and 4% when $|C|$ is equal to five, 10 and 15, respectively. Looking at Table 7, we observe that the VNS is very effective on instances belonging to the classes C1 and RC1: on average the gap is about 2%.

Results on large-size instances Since CPLEX does not find optimal solutions for large-size instances and requires considerable time to find feasible solutions, we solve the large-size instances with the VNS and we make comparisons, when possible, with the CPLEX performance. The computational values are obtained by setting $k_{\max} = 50$ and $improve = 50$ for the VNS. We first compare the solutions obtained by solving the problems with the VNS, with the first feasible solution found by CPLEX within the imposed time limit.

Table 8 summarizes the average results over the large-size instances solved by both the VNS and CPLEX, grouped by class and C size. In the first column we report the class, in the second one the size $|C|$. The third column shows the average time [sec] required by the VNS and the fourth one the average speedup. The fifth column provides the average value of the objective function of the solution found by the VNS (Obj VNS), in the sixth column we show the percentage gap in cost (Gap_c). Finally, since CPLEX does not always find a feasible solution within the imposed time limit, we report the percentage of instances solved by CPLEX referred to as %slv.

The results summarized in Table 8 show that the VNS clearly outperforms CPLEX in terms of both effectiveness and efficiency. For the instances belonging to the class C1, the VNS is more than four times faster than CPLEX for instances with $|C| = 30$, and up to about five times faster when $|C|$ is 40 and 50. In addition, CPLEX finds solutions for all instances when $|C| = 30$ with a Gap_c equal to -76%, and for 60% of all instances only when $|C| = 40$ and $|C| = 50$ with a gap equal to -79% and -80%, respectively.

We observe on average a similar trend for the instances of classes R1 and RC1. The VNS is more efficient than CPLEX for all the instances belonging to the class R1. In particular, when $|C| = 40$ the VNS is more than 33 times faster than CPLEX. The VNS is also more effective, the value of Gap_c is about -65%, -58% and -87% for the instances with $|C|$ equal to 30, 40 and 50, respectively. In addition, with the only exception of the instances with $|C| = 30$, CPLEX is not able to find a feasible solution for all the instances; indeed when $|C| = 50$ it solves only the 20% of them.

For the instances of class RC1, CPLEX is able to find a feasible solution for all the instances with $|C| = 30$ and $|C| = 40$, but the VNS is still more effective than CPLEX; indeed the gap is -77% and -26%, respectively.

In order to investigate the performance of our VNS, we carry out a second phase of tests on large-size instances. We impose a new time limit on CPLEX equal to the time required by the VNS to find the solution. Table 9 summarizes the average results over the instances solved by both the VNS and CPLEX. The first column shows the class, the second one the number of customers belonging to the set C, the third column reports the time spent by the VNS. The fourth column provides the VNS objective function, the fifth one the gap and the last one the percentage number of instances solved by CPLEX. The results clearly show that the VNS is more effective.

CPLEX is able to find a feasible solution for all the considered instances only when $|C| = 30$

of the classes C1 and RC1, with a gap of about -76.5% . For the instances of the class R1, CPLEX finds a feasible solution for only the 20% and 40% of the instances when $|C| = 30$ and $|C| = 40$, with a gap of -83% and -85%, respectively, and it does not find solutions for any instance when $|C| = 50$. Overall, the VNS clearly outperforms CPLEX in terms of effectiveness.

5 Conclusions

We have introduced, modeled, and solved a new crowd-shipping variant. We have considered the presence of a single depot, at which vehicles start and end their routes, and several transshipment nodes which act as intermediate depots. The occasional drivers can decide to serve some customers by picking up the parcels from either the central depot or the transshipment nodes. Starting from the central depot, the vehicles serve customers and transshipment nodes. The parcels delivered by the occasional drivers must be available at the transshipment nodes chosen by them.

We have developed a mathematical model as well as a variable neighbourhood search metaheuristic which was extensively tested on modified Solomon instances. Our results highlight several trends, in terms of the organization of the deliveries, considering different parameters of the problem and show the benefits of introducing transshipment nodes in the service network. We compared the solutions determined by CPLEX with those obtained by the proposed metaheuristic. Our results show a good behaviour of the proposed metaheuristic with a reasonable optimality gap within short computational times.

References

[1] Y. Amarouche, R. N. Guibadj, and A. Moukrim. A neighborhood search and set cover hybrid heuristic for the two-echelon vehicle routing problem. In Ralf Borndörfer and Sabine Storandt, editors, *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*, volume 11 of *Open Access Series in Informatics*, pages 1–15, Helsinki; Finland, 2018. Dagstuhl Publishing,.

[2] A. Anderluh, V. C. Hemmelmayr, and P. C. Nolz. Synchronizing vans and cargo bikes in a city distribution network. *Central European Journal of Operations Research*, 25:345–376, 2017.

[3] C. Archetti, M. W. P. Savelsbergh, and M. G. Speranza. The vehicle routing problem with occasional drivers. *European Journal of Operational Research*, 254:471–480, 2016.

[4] A. M. Arslan, N. Agatz, L. Kroon, and R. Zuidwijk. Crowdsourced delivery: A dynamic pickup and delivery problem with ad-hoc drivers. Technical report, ERIM, Report Series Reference, 2016.

[5] A. Barr and J. Wohl. Exclusive: Wal-Mart may get customers to deliver packages to online buyers. REUTERS - Business, 2013.

[6] O. Belgina, I. Karaoglanb, and F. Altiparmakc. Two-echelon vehicle routing problem with simultaneous pickup and delivery: Mathematical model and heuristic approach. *Computers & Industrial Engineering*, 115:1–16, 2018.

[7] G. Bensinger. Amazon's next delivery drone: You. Wall Street Journal, 2015.

[8] H. Buldeo Rai, S. Verlinde, J. Merckx, and C. Macharis. Crowd logistics: an opportunity for more sustainable urban freight transport? *European Transport Research Review*, 9(3):39, 2017.

[9] T.G. Crainic, N. Ricciardi, and G. Storchi. Models for evaluating and planning city logistics systems. *Transportation Science*, 43(4):432–454, 2009.

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

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

[12] W. Jie, J. Yang, M. Zhang, and Y. Huang. The two-echelon capacitated electric vehicle routing problem with battery swapping stations: Formulation and efficient methodology. *European Journal of Operational Research*, 272:879–904, 2019.

[13] R. Landa. *Thinking Creatively in the Digital Age*. Nimble, Blue Ash, Ohio, 1^{st} edition, 2015.

[14] G. Laporte and Y. Nobert. A vehicle flow model for the optimal design of a two-echelon distribution system. In H.A. Eiselt and G. Pederzoli, editors, *Advances in Optimization and Control*, volume 302 of *Lecture Notes in Economics and Mathematical Systems*, pages 158–173, Berlin, Heidelberg, 1988. Springer.

[15] G. Macrina, L. Di Puglia Pugliese, F. Guerriero, and D. Laganà. The vehicle routing problem with occasional drivers and time windows. In A. Sforza and C. Sterle, editors, *Optimization and Decision Science: Methodologies and Applications*, volume 217 of Springer Proceedings in Mathematics & Statistics, pages 577–587, Cham, Switzerland, 2017. ODS, Sorrento, Springer.

[16] G. Perboli and R. Tadei. New families of valid inequalities for the two-echelon vehicle routing problem. *Electronic Notes in Discrete Mathematics*, 36:639–646, 2010.

[17] G. Perboli, R. Tadei, and D. Vigo. The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. *Transportation Science*, 45(3):364–380, 2011.

[18] T. Raviv and E. Z. Tenzer. Crowd-shipping of small parcels in a physical Internet. https://www.researchgate.net/publication/326319843_Crowd-shipping_of_small_parcels_in_a_physical_internet, July 2018.

[19] M. W. P. Savelsbergh and T. Van Woensel. City logistics: challenges and opportunities. *Transportation Science*, 50(2):579–590, 2016.

[20] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265, 1987.

[21] L. Zhou, R. Baldacci, D. Vigo, and X. Wang. A multi-depot two-echelon vehicle routing problem with delivery options arising in the last mile distribution. *European Journal of Operational Research*, 265:765–778, 2018.

			Time	Obj	Z_{cv}	Z_k	Ts	#vs	#ODs	%ODsD	%vsd
$\|C\| = 5$	$\|T\|$	2	0.95	3132.69	394.64	2738.05	2.00	2.20	32.79	28%	44%
		4	4.08	2772.72	424.16	2348.56	3.82	2.52	32.96	13%	41%
		6	13.81	2790.69	420.13	2370.57	4.81	2.69	32.94	11%	41%
	α	0.7	8.05	3152.75	433.36	2719.39	3.37	2.79	33.16	32%	37%
		1.0	9.31	2898.37	435.91	2462.46	3.57	2.68	32.95	17%	41%
		1.3	4.49	2798.75	399.69	2399.05	3.61	2.41	32.84	12%	43%
		2.0	3.25	2744.94	382.93	2362.00	3.62	2.00	32.64	8%	47%
	β	1.0	8.84	2926.24	416.28	2509.96	3.62	2.46	32.84	19%	43%
		1.3	5.60	2891.54	414.49	2477.05	3.56	2.47	32.89	17%	42%
		2.0	4.39	2878.32	408.15	2470.16	3.46	2.48	32.95	16%	41%
	ρ	0.75	4.66	1651.67	364.45	1287.22	3.42	2.28	33.42	19%	32%
		1.50	6.40	2905.07	426.19	2478.89	3.61	2.48	32.74	17%	45%
		2.25	7.77	4139.36	448.28	3691.07	3.59	2.64	32.53	16%	49%
	Average		6.28	2898.70	412.97	2485.73	3.54	2.47	32.90	17%	42%
$\|C\| = 10$	$\|T\|$	2	2.25	3165.03	469.94	2695.08	2.00	2.18	35.73	30%	43%
		4	10.34	2804.01	451.35	2352.66	3.87	2.49	36.88	17%	31%
		6	29.81	2802.80	454.41	2348.39	4.81	2.64	36.92	14%	31%
	α	0.7	17.35	3148.54	484.67	2663.86	3.42	2.79	37.01	33%	30%
		1.0	20.41	2909.22	478.13	2431.08	3.59	2.70	36.39	19%	36%
		1.3	10.08	2837.24	440.28	2396.95	3.61	2.30	36.39	16%	36%
		2.0	8.70	2800.79	431.18	2369.61	3.61	1.98	36.25	14%	37%
	β	1.0	18.40	2938.93	466.74	2472.19	3.64	2.43	36.35	21%	37%
		1.3	13.35	2920.08	458.97	2461.11	3.59	2.44	36.52	20%	35%
		2.0	10.65	2912.82	449.99	2462.83	3.44	2.45	36.66	20%	33%
	ρ	0.75	16.96	1682.29	393.34	1288.95	3.48	2.26	37.43	22%	26%
		1.50	11.17	2933.46	466.13	2467.33	3.59	2.45	36.36	20%	36%
		2.25	14.28	4156.08	516.23	3639.85	3.60	2.61	35.74	19%	43%
	Average		14.13	2923.94	458.57	2465.38	3.56	2.44	36.51	20%	35%
$\|C\| = 15$	$\|T\|$	2	10.52	3208.62	578.47	2630.15	2.00	2.25	37.32	29%	51%
		4	24.54	2852.40	524.79	2327.62	3.89	2.49	39.13	18%	39%
		6	78.11[2]	2851.12	521.41	2329.70	4.79	2.60	39.28	15%	38%
	α	0.7	62.88[1]	3213.76	584.08	2629.68	3.41	2.88	39.02	33%	40%
		1.0	48.15[1]	2954.56	553.97	2400.59	3.61	2.73	38.82	19%	41%
		1.3	21.55	2871.71	515.12	2356.59	3.60	2.23	38.32	16%	45%
		2.0	18.30	2842.82	513.05	2329.77	3.62	1.96	38.14	13%	46%
	β	1.0	49.96[2]	2983.56	546.61	2436.95	3.62	2.44	38.48	21%	43%
		1.3	32.56	2967.32	541.81	2425.51	3.58	2.45	38.57	20%	43%
		2.0	30.64	2961.26	536.25	2425.02	3.49	2.46	38.68	20%	42%
	ρ	0.75	31.87	1735.75	452.33	1283.42	3.50	2.28	40.19	23%	32%
		1.50	31.28	2986.77	545.63	2441.14	3.60	2.48	38.54	20%	43%
		2.25	50.01[2]	4189.61	626.71	3562.91	3.59	2.58	36.99	18%	53%
	Average		37.72[2]	2970.71	541.56	2429.16	3.56	2.45	38.58	20%	43%

Table 3: Average computational results for the instances R1.

			Time	Obj	Z_{cv}	Z_k	Ts	#vs	#ODs	%ODsD	%vsd
$\|C\| = 5$	$\|T\|$	2	1.93	4654.52	332.56	4321.96	1.37	1.61	33.14	56%	37%
		4	6.41	3806.78	354.24	3452.54	2.41	2.20	33.34	26%	33%
		6	29.15	3933.26	356.02	3577.24	3.00	2.08	33.38	31%	32%
	α	0.7	12.36	4246.33	349.79	3896.55	2.17	2.27	33.39	46%	32%
		1.0	8.01	4110.70	347.41	3763.30	2.27	2.06	33.29	36%	34%
		1.3	10.97	4094.31	349.01	3745.30	2.28	1.89	33.25	35%	35%
		2.0	18.65	4074.73	344.21	3730.52	2.31	1.63	33.21	34%	36%
	β	1.0	13.54	4219.06	346.84	3872.23	2.36	1.89	33.28	42%	34%
		1.3	13.10	4138.46	347.39	3791.06	2.32	1.96	33.31	38%	34%
		2.0	10.85	4037.04	348.58	3688.46	2.09	2.04	33.27	34%	35%
	ρ	0.75	12.42	2211.97	233.84	1978.13	1.83	1.54	34.22	44%	16%
		1.50	14.49	4150.73	372.13	3778.60	2.37	2.08	33.08	36%	38%
		2.25	10.58	6031.86	436.84	5595.02	2.57	2.27	32.56	33%	49%
	Average		12.49	4131.52	347.60	3783.92	2.26	1.96	33.29	38%	34%
$\|C\| = 10$	$\|T\|$	2	5.02	4653.52	390.82	4262.69	1.35	1.80	35.19	58%	48%
		4	19.40	3751.53	418.64	3332.89	2.29	2.33	35.22	28%	48%
		6	122.84^{10}	3881.63	415.11	3466.52	2.84	2.17	35.53	33%	45%
	α	0.7	45.37^{2}	4200.87	415.48	3785.39	2.10	2.44	35.41	47%	46%
		1.0	38.00^{1}	4081.21	413.68	3667.53	2.24	2.21	35.34	38%	47%
		1.3	48.10^{2}	4058.57	406.42	3652.16	2.19	2.01	35.26	37%	47%
		2.0	64.89^{5}	4041.58	397.19	3644.39	2.12	1.74	35.24	36%	48%
	β	1.0	45.84^{3}	4183.41	408.10	3775.31	2.31	2.04	35.34	43%	47%
		1.3	58.54^{5}	4101.57	407.62	3693.94	2.20	2.08	35.36	40%	46%
		2.0	42.89^{2}	4001.70	408.85	3592.85	1.97	2.18	35.23	36%	48%
	ρ	0.75	43.35^{3}	2219.14	258.42	1960.73	1.81	1.62	37.79	47%	22%
		1.50	59.61^{5}	4125.22	407.19	3718.03	2.28	2.04	35.40	38%	46%
		2.25	44.31^{2}	5942.31	558.97	5383.34	2.39	2.63	32.74	32%	73%
	Average		49.00^{10}	4095.56	408.19	3687.37	2.16	2.10	35.31	40%	47%
$\|C\| = 15$	$\|T\|$	2	25.03	4686.17	455.67	4230.50	1.45	1.92	38.80	59%	41%
		4	98.23^{5}	3714.13	467.92	3246.21	2.40	2.37	39.09	29%	39%
		6	388.40^{67}	3848.58	478.91	3369.67	3.11	2.34	38.96	34%	40%
	α	0.7	203.94^{22}	4195.17	461.51	3733.67	2.23	2.57	39.61	48%	36%
		1.0	140.72^{15}	4064.35	467.60	3596.75	2.39	2.25	38.89	39%	41%
		1.3	169.59^{18}	4040.60	474.57	3566.03	2.33	2.12	38.63	38%	42%
		2.0	167.98^{17}	4031.72	466.33	3565.39	2.33	1.90	38.68	37%	42%
	β	1.0	169.19^{23}	4170.66	467.72	3702.94	2.44	2.17	38.97	44%	40%
		1.3	170.89^{24}	4087.56	470.42	3617.14	2.34	2.21	38.91	41%	41%
		2.0	171.59^{25}	3990.67	464.37	3526.30	2.18	2.26	38.98	37%	40%
	ρ	0.75	143.10^{18}	2243.30	286.95	1956.36	1.92	1.73	42.09	47%	19%
		1.50	166.08^{23}	4114.94	496.36	3618.58	2.49	2.26	38.72	39%	42%
		2.25	202.49^{31}	5890.64	619.20	5271.44	2.55	2.64	36.04	34%	60%
	Average		170.55^{72}	4082.96	467.50	3615.46	2.32	2.21	38.95	40%	40%

Table 4: Average computational results for the instances C1.

			Time	Obj	Z_{cv}	Z_k	Ts	#vs	#ODs	%ODsD	%vsd
$\|C\| = 5$	$\|T\|$	2	0.79	4200.66	313.08	3887.59	1.90	1.85	33.73	44%	25%
		4	2.48	3883.47	338.91	3544.56	3.14	2.57	33.70	17%	26%
		6	6.83	3704.80	354.38	3350.42	3.76	2.43	33.75	21%	25%
	α	0.7	2.93	4006.96	326.79	3680.17	2.80	2.51	34.00	38%	20%
		1.0	2.67	3915.61	340.08	3575.53	2.96	2.47	33.65	25%	27%
		1.3	3.23	3903.25	341.29	3561.96	2.99	2.33	33.63	23%	27%
		2.0	4.64	3892.77	333.67	3559.10	2.99	1.83	33.63	23%	27%
	β	1.0	4.33	3972.49	342.37	3630.12	3.17	2.22	33.68	30%	26%
		1.3	3.07	3930.18	333.56	3596.62	2.91	2.28	33.75	27%	25%
		2.0	2.70	3886.27	330.44	3555.83	2.72	2.36	33.75	24%	25%
	ρ	0.75	3.53	2121.84	260.92	1860.92	2.48	2.01	34.33	31%	13%
		1.50	3.33	3940.10	345.18	3594.93	3.11	2.40	33.90	26%	22%
		2.25	3.24	5727.00	400.27	5326.73	3.21	2.44	32.95	24%	41%
	Average		3.37	3929.65	335.46	3594.19	2.93	2.29	33.73	27%	25%
$\|C\| = 10$	$\|T\|$	2	2.07	4214.44	368.03	3846.41	1.93	1.90	36.71	45%	33%
		4	5.15	3879.16	418.47	3460.69	3.40	2.66	36.38	17%	36%
		6	19.42	3695.57	429.41	3266.16	3.97	2.50	36.54	22%	35%
	α	0.7	10.20	4017.52	387.77	3629.75	2.99	2.61	37.59	38%	24%
		1.0	6.64	3911.30	413.22	3498.07	3.13	2.58	36.36	26%	36%
		1.3	7.30	3900.41	413.23	3487.18	3.12	2.35	36.12	25%	39%
		2.0	11.39	3889.66	406.98	3482.68	3.16	1.87	36.10	24%	39%
	β	1.0	12.38	3969.63	411.80	3557.83	3.31	2.29	36.41	31%	36%
		1.3	7.16	3926.07	404.90	3521.17	3.08	2.36	36.53	28%	35%
		2.0	7.11	3893.46	399.20	3494.26	2.92	2.40	36.68	26%	33%
	ρ	0.75	11.39	2157.84	309.92	1847.92	2.79	2.15	38.83	32%	12%
		1.50	6.91	3943.45	425.47	3517.99	3.17	2.43	36.03	27%	40%
		2.25	8.35	5687.87	480.51	5207.36	3.34	2.47	34.77	25%	52%
	Average		8.88	3929.72	405.30	3524.42	3.10	2.35	36.54	28%	35%
$\|C\| = 15$	$\|T\|$	2	8.34	4194.72	428.03	3766.69	2.00	2.07	39.32	44%	38%
		4	22.69	3831.67	458.83	3372.85	3.56	2.69	39.72	18%	35%
		6	81.74[4]	3628.95	468.75	3160.20	4.02	2.62	39.78	21%	35%
	α	0.7	51.67[3]	3976.26	443.35	3532.91	3.11	2.76	40.53	38%	30%
		1.0	25.61	3864.73	458.92	3405.81	3.21	2.63	39.36	26%	38%
		1.3	30.02	3854.88	458.23	3396.65	3.24	2.44	39.29	24%	38%
		2.0	43.06[1]	3844.59	446.97	3397.62	3.21	2.00	39.24	23%	38%
	β	1.0	39.90[1]	3920.11	457.90	3462.22	3.36	2.43	39.51	30%	37%
		1.3	39.11[1]	3880.20	450.31	3429.90	3.16	2.46	39.64	28%	36%
		2.0	33.76[2]	3855.03	447.40	3407.63	3.07	2.48	39.67	25%	36%
	ρ	0.75	39.61[1]	2156.17	353.26	1802.91	3.03	2.37	42.29	30%	18%
		1.50	32.71[1]	3898.92	462.39	3436.53	3.23	2.46	39.26	27%	38%
		2.25	40.44[2]	5600.25	539.95	5060.30	3.31	2.54	37.27	25%	52%
	Average		37.59[4]	3885.11	451.87	3433.25	3.19	2.46	39.61	28%	36%

Table 5: Average computational results for the instances RC1.

		speedup	Gap_c
$\|C\| = 5$	C1	21.94	0.98%
	R1	8.58	3.50%
	RC1	6.97	1.07%
Average		12.50	1.85%
$\|C\| = 10$	C1	41.72	2.29%
	R1	13.85	4.82%
	RC1	14.66	2.12%
Average		23.41	3.08%
$\|C\| = 15$	C1	115.48	2.93%
	R1	31.21	6.16%
	RC1	45.38	3.91%
Average		64.02	4.33%

Table 6: Computational results for small-size instances.

	speedup	Gap_c
C1	59.71	2.06%
R1	17.88	4.83%
RC1	22.33	2.36%

Table 7: Computational results for small-size instances, grouped by class.

Class	$\|C\|$	Time VNS	speedup	Obj VNS	gap_c	%slv
C1	30	3.99	4.04	4174.44	−76%	100%
	40	3.59	4.79	3018.93	−79%	60%
	50	10.47	4.89	4396.80	−80%	60%
R1	30	3.89	7.19	3168.51	−65%	100%
	40	3.44	33.05	2254.94	−58%	60%
	50	3.16	13.27	1550.43	−87%	20%
RC1	30	3.86	0.00	4019.61	−77%	100%
	40	7.43	38.31	4344.67	−26%	100%
	50	6.22	5.92	3150.39	−80%	60%

Table 8: Computational results for large-size instances: VNS against CPLEX first feasible solution.

Class	$\|C\|$	Time VNS	Obj VNS	Gap_c	%slv
C1	30	3.99	4174.44	−76%	100%
	40	4.29	3549.84	−79%	40%
	50	10.47	4396.80	−80%	60%
R1	30	0.75	1382.16	−83%	20%
	40	3.33	1893.05	−85%	40%
	50	-	-	-	0%
RC1	30	3.86	4019.61	−77%	100%
	40	2.11	2039.74	−82%	20%
	50	4.53	3191.39	−81%	20%

Table 9: Computational results for large-size instances: VNS against CPLEX imposing the heuristic time as time limit.