A Variable Neighborhood Search for the Vehicle Routing Problem with Occasional Drivers and Time Windows

Giusy Macrina, Luigi Di Puglia Pugliese and Francesca Guerriero
Department of Mechanical, Energy and Management Engineering, University of Calabria, 87036 Rende (CS), Italy

Keywords: Vehicle Routing, Crowd-shipping, Variable Neighborhood Search.

Abstract: This paper presents a Variable Neighborhood Search algorithm for a Vehicle Routing Problem variant with a crowd-sourced delivery policy. We consider a heterogeneous fleet composed of conventional capacitated vehicles and some ordinary drivers, called occasional drivers, who accept to deviate from their route to deliver items to other people in exchange for a small compensation. The objective is to minimize total costs, that is conventional vehicles costs plus occasional drivers compensation. Our computational study shows that the Variable Neighborhood Search is highly effective and able to solve large-size instances within short computational times.

1 INTRODUCTION

During the last years, the exponential growth of e-commerce has lead to important changes in world's economy landscape. E-commerce offers several advantages, such as the possibility to buy anything, anywhere, anytime, to find products that are not present in physical markets or at a low prices spending overall less money and time. For these reasons, people increasingly use the Internet to buy goods. In 2021 retail revenues are projected to grow to 4.88 trillion US dollars, that is about the 112% of the sales worldwide registered in 2017 (eMarketer, 2018). In this context, the main challenge for the companies is to provide a more efficient distribution service to more demanding customers. Several big e-retailers have started to look for innovative and more effective transportation solutions to speed up the last-mile and same-day deliveries. Crowd-shipping is one of the most innovative solution, among the others. The main idea of crowd-shipping is to apply the concept of "sharing economy" to the deliveries, by connecting customers who need to receive a package, to other people who have space in their own car and accept to make a deviation from their route to perform the delivery, for a small compensation. These temporary couriers are named Occasional Drivers (ODs). Several big companies, such as Walmart (see Barr & Wohl (Barr and Wohl, 2013)), DHL (see Landa (Landa, 2015)) and Amazon (see Besinger (Bensinger, 2015)) have started to adopt this solution, by implementing and using a crowd-sourcing platform to connect customers to ODs. The success of crowd-shipping is related to several benefits that it offers. It does not require any infrastructure, it is flexible, thus it can offer a more customized service, it is less expensive than traditional delivery policy and it can reduce environmental impacts by reducing the number of vehicles on the roads. For a literature review of the recent contributions on crowd-shipping the reader is referred to Arslan et al. (Arslan et al., 2016).

Archetti et al. (Archetti et al., 2016) introduced the vehicle routing problem with occasional drivers (VRPOD). In the VRPOD the fleet is composed of traditional capacitated vehicles and ODs. Each OD can serve at most one customer. They modelled and solved the VRPOD with a multi-start heurisic which combines tabu search and Variable Neighborhood Search (VNS). In addition, the authors performed several tests to show how different ODs' compensation schemes may influence the delivery plan. Macrina et al. (Macrina et al., 2017) extended this work by introducing the time windows for both customers and ODs and allowing for multiple deliveries. They carried out computational tests to show the overall benefits of using crowd-shipping and multiple deliveries. Finally, they studied the advantages of applying split delivery policy for ODs deliveries. The authors proposed mathematical formulations and analyze the optimal solutions by considering several scenarios. The aim of Macrina et al. (Macrina et al., 2017) was to highlight the potential of considering multiple

Macrina, G., Pugliese, L. and Guerriero, F.
A Variable Neighborhood Search for the Vehicle Routing Problem with Occasional Drivers and Time Windows.
DOI: 10.5220/0009193302700277
In *Proceedings of the 9th International Conference on Operations Research and Enterprise Systems (ICORES 2020)*, pages 270-277
ISBN: 978-989-758-396-4; ISSN: 2184-4372

deliveries and split policy for the ODs. The authors studied the behaviour of the transportation system by solving to optimality small-size networks. They obtained optimal solution for instances up to 15 customers. The work of Macrina et al. (Macrina et al., 2017) was recently extended by Macrina et al. (Macrina et al., 2020), who introduced the transshipment nodes in the service network. Transshipment nodes are closer than the main depot to the delivery area, they are intermediate depots for ODs and have to be served by the classical vehicle. The authors formulated the problem as a particular instance of the two-echelon VRP and proposed a math-heuristic algorithm based on the VNS framework. Dahle et al. (Dahle et al., 2019) introduced the pickup and delivery problem with time windows and ODs, an extension of the problem proposed in Archetti et al. (Archetti et al., 2016). They modeled the problem, than studied the impact of introducing ODs in transportation planning, focusing on different compensations schemes. Their computational study highlighted the benefits of employing the ODs, even if the suboptimal compensation scheme is used. Macrina and Guerriero (Macrina and Guerriero, 2018) modeled the green vehicle routing problem with ODs, in which a fleet composed of classical engine fuelled vehicles, electrical vehicles and ODs is used for the deliveries. They focused on the environmental impacts of the three types of vehicles and concluded that the joint use of electrical vehicles and ODs may reduce the overall polluting emissions.

All the mentioned contributions suppose to operate in a static system. Thus, all the information related to the demands and the availability of the ODs are known a priori. Recently, several researchers have started to consider a more realistic framework in a dynamic context. Dahle et al. (Dahle et al., 2017) considered the availability of ODs as an uncertain parameter and supposed to know some stochastic information about their appearance. They proposed a stochastic model and compared its performance with those obtained by using deterministic strategies with re-optimization. They highlighted that the use of a stochastic formulation may lead to more profitable solutions in terms of saving costs. (Dayarian and Savelsbergh, 2017) considered a VRPOD variant with dynamic customers requests. They developed and compared two rolling horizon dispatching approaches: a myopic one that ignores all the information regarding future events and one that considers probabilistic information about future on-line order and ODs appearance. In their dynamic VRPOD variant, Gdowska et al. (Gdowska et al., 2018) supposed that the ODs may reject assignments and gave to each pair (OD-request) a random probability. They proposed a bi-level stochastic formulation of the problem and a heuristic approach to solve it. Their results pointed out the needs of defining a dynamic compensation scheme for ODs. All these contributions clearly highlight the interesting results that could be obtained with the crowd-shipping, not only in terms of reduction of operational costs, but also in terms of reduction of environmental impacts and optimization of the quality of service. Thus, the majority of scientific contributions focused on the study of the impacts of using ODs in transportation planning. However, scarce attention has been devoted to the implementation of effective and efficient solution approaches for this problem.

In this paper, we propose a metaheuristic to address the more general static version of the VRPOD with time windows and multiple deliveries (VRPODTWmd) proposed by Macrina et al. (Macrina et al., 2017). In their work Macrina et al. (Macrina et al., 2017) solved only small-size instances, thus we want to fill this gap by proposing a VNS heuristic which solves more sized instances. We carry out several computational tests on different size networks and we assess the effectiveness of the proposed algorithm. Since we compare the results obtained with the VNS heuristic with the optimal solution, for the sake of completeness, we report in this paper the formulation proposed in (Macrina et al., 2017). It is important to highlight that our solution approach can be easily adapted to handle several variants of the VRPOD, arising either in static or dynamic settings.

The remainder of the paper is organized as follows. In Section 2 we report the mathematical formulation of the VRPODTWmd proposed by Macrina et al. (Macrina et al., 2017). In Section 3 we describe our VNS heuristic for the VRPODTWmd. In Section 4 we describe the computational experiments and we analyze the numerical results. In Section 5 we summarize the conclusions.

2 THE VEHICLE ROUTING PROBLEM WITH OCCASIONAL DRIVERS AND TIME WINDOWS

In this section we describe the mathematical formulation of the VRPODTWmd proposed by Macrina et al. (Macrina et al., 2017). Let C be the set of customers, let s be the origin node and t the destination node for the classical vehicles, i.e., those belonging to the company. Let K be the set of available ODs and V

the set of v_k destinations associated with the ODs. We define the node set as $N = C \cup \{s,t\} \cup V$. We model the problem on a complete directed graph $G = (N,A)$, where A is the set of arcs. Each arc $(i,j) \in A$ has a cost c_{ij} and a travel time t_{ij} associated with it. Note that both c_{ij} and t_{ij} satisfy the triangle inequality. Each node $i \in C \cup V$ has a time window $[e_i, l_i]$, and each customer $i \in C$ has a demand d_i. Q is the capacity of the classical vehicles, P is the number of available classical vehicles and Q_k is the capacity of OD $k \in K$. Let x_{ij} be a binary variable equal to 1 if and only if a classical vehicle traverses arc (i,j). For each node $i \in N$ let y_i be the available capacity of a classical vehicle after visiting customer i, and let s_i be the arrival time of a classical vehicle at customer i. Moreover r^k_{ij} is a binary variable equal to 1 if and only if OD $k \in K$ traverses arc (i,j). Let f^k_i indicate the arrival time of OD k at customer i, and let w^k_i be the available capacity of the vehicle associated with OD k after visiting customer i. Table 1 summarizes the notation.

The objective function for the VRPODTWmd, inspired by Archetti et al. (Archetti et al., 2016), is as follows:

$$Min\, obj_c = \sum_{i \in C \cup \{s\}} \sum_{j \in C \cup \{t\}} c_{ij} x_{ij} + \sum_{k \in K} \sum_{i \in C \cup \{s\}} \sum_{j \in C} \rho c_{ij} r^k_{ij} - \sum_{k \in K} \sum_{j \in C} c_{sv_k} r^k_{sj}, \quad (1)$$

which minimizes the total costs. The first term of obj_c is the transportation cost associated with classical vehicles. The second term is the compensation cost of OD k for the delivery service, where $\rho \geq 0$ is the compensation factor for the ODs, the third one is the cost of OD k when it does not perform the delivery service (i.e. we pay the OD k only for the deviation, thus we do not pay the portion of the road from the origin to the destination).

The constraints of VRPODTWmd are as follows:

$$\sum_{j \in C \cup \{t\}} x_{ij} - \sum_{j \in C \cup \{s\}} x_{ji} = 0 \qquad i \in C \quad (2)$$

$$\sum_{j \in C} x_{sj} - \sum_{j \in C} x_{jt} = 0 \quad (3)$$

$$y_j \geq y_i + d_j x_{ij} - Q(1 - x_{ij}) \qquad j \in C \cup \{t\}, i \in C \cup \{s\} \quad (4)$$

$$y_s \leq Q \quad (5)$$

$$s_j \geq s_i + t_{ij} x_{ij} - \alpha(1 - x_{ij}) \qquad i \in C, j \in C \quad (6)$$

$$e_i \leq s_i \leq l_i \qquad i \in C \quad (7)$$

$$\sum_{j \in C} x_{sj} \leq P \quad (8)$$

$$\sum_{j \in C \cup \{v_k\}} r^k_{ij} - \sum_{h \in C \cup \{s\}} r^k_{hi} = 0 \qquad i \in C, k \in K \quad (9)$$

$$\sum_{j \in C \cup \{v_k\}} r^k_{sj} - \sum_{j \in C \cup \{s\}} r^k_{jv_k} = 0 \qquad k \in K \quad (10)$$

$$\sum_{k \in K} \sum_{j \in C \cup \{v_k\}} r^k_{sj} \leq |K| \quad (11)$$

$$\sum_{j \in C} r^k_{sj} \leq 1 \qquad k \in K \quad (12)$$

$$w^k_j \geq w^k_i + d_i r^k_{ij} - Q_k(1 - r^k_{ij}) \qquad j \in C \cup \{v_k\}, \; i \in C \cup \{s\}, k \in K \quad (13)$$

$$w^k_s \leq Q_k \qquad k \in K \quad (14)$$

$$f^k_i + t_{ij} r^k_{ij} - \alpha(1 - r^k_{ij}) \leq f^k_j \qquad i \in C, j \in C, k \in K \quad (15)$$

$$f^k_i \geq e_{v_k} + t_{si} \qquad i \in C, k \in K \quad (16)$$

$$f^k_{v_k} \leq l_{v_k} \qquad k \in K \quad (17)$$

$$f^k_i + t_{iv_k} r^k_{iv_k} - \alpha(1 - r^k_{iv_k}) \leq f^k_{v_k} \qquad i \in C, k \in K \quad (18)$$

$$e_i \leq f^k_i \leq l_i \qquad i \in C \quad (19)$$

$$\sum_{j \in C \cup \{t\}} x_{ij} + \sum_{h \in C \cup \{v_k\}} \sum_{k \in K} r^k_{ih} = 1 \qquad i \in C \quad (20)$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A \quad (21)$$

$$r^k_{ij} \in \{0,1\} \qquad (i,j) \in A, k \in K \quad (22)$$

$$0 \leq y_i \leq Q \qquad i \in C \cup \{s,t\} \quad (23)$$

$$0 \leq w^k_i \leq Q_k \qquad i \in C \cup \{s, v_k\}, k \in K \quad (24)$$

$$f^k_i \geq 0 \qquad i \in C \cup \{s, v_k\}, k \in K. \quad (25)$$

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

s	origin node
t	destination node for classical vehicles
C	set of customers
K	set of available occasional drivers
V	set of v_k destinations for the occasional drivers
A	set of arcs
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
α	large number
Q	capacity of classical vehicles
P	number of classical vehicles
Q_k	capacity of occasional driver k
x_{ij}	binary decision variable indicating if arc $(i,j) \in A$ is traversed by a classical vehicle
y_i	decision variable specifying the available capacity of the classical vehicle after visiting customer i
s_i	decision variable specifying the arrival time of the classical vehicle to customer i
r^k_{ij}	binary decision variable indicating if arc $(i,j) \in A$ is traversed by the occasional driver k
f^k_i	decision variable specifying the arrival time of the occasional driver k at customer i
w^k_i	decision variable specifying the available capacity of the occasional driver k after visiting customer i

Constraints (2) to (8) are linked to the classical vehicles. In particular, constraints (2) and (3) are the flow conservation constraints. Conditions (4) and (5) represent the capacity constraints. Constraints (6) allow the determination of the arrival time at node j, while conditions (7) represent the time windows constraints. Constraint (8) imposes a maximum number of available vehicles. Constraints (9) to (19) are linked to the ODs. Conditions (9) and (10) are the

flow conservation constraints. Constraints (11) and (12) impose a limit on the number of available ODs and on the number of departures from the depot, respectively. Conditions (13) and (14) are the capacity constraints. Constraints (15) compute the arrival time at node j. Conditions (16) and (17) are the time window constraints and also define the time at which the ODs are available to make deliveries, while constraints (18) compute the arrival time at the destination node v_k. Constraints (19) guarantee that each customer is served within its time window. Constraints (20) mean that each customer is visited at most once, either by a classical vehicle or by an OD. Constraints (21) to (25) define the domains of variables.

3 VARIABLE NEIGHBORHOOD SEARCH

In this section we describe our VNS for the VRPODTWmd. Algorithm 1 presents the VNS scheme. N^h, for $h = 0, ..., h_{max}$ represents the set of neighborhoods and k_{max} is the maximum number of iterations. We generate an initial solution δ, then we apply a *Shaking* phase to perturb δ. The result of *Shaking* phase δ' is the starting point for the local search procedure, in particular we use a Variable Neighborhood Descent (VND) for obtaining a new solution δ''. If the generated solution δ'' is more effective than δ (i.e. the value of the objective function f(δ'') is lower than f(δ')), it replaces δ and $h = 0$ otherwise we increment h. The algorithm terminates when either all the neighborhoods in N^h are explored or when the number of iterations reaches the value h_{max}.

Algorithm 1: Variable neighborhood search.

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

Initial Solution. For obtaining an initial solution we use an insertion heuristic, adapted for the VRPODTWmd. The starting tour is composed of the origin and destination nodes. The heuristic inserts a new node in the tour in the best feasible position. At first, we try to serve the farthest customers with ODs and then the unserved customers with traditional drivers. Since the initial solution may be infeasible, we apply a repair phase in which the LS moves are applied until a feasible solution is generated.

Local Search Moves. We generate the neighborhoods by using seven different Local Search (LS) moves:

1. 2-Opt: This operator removes two arcs (i, j) and (u, v) in the same route r (classical or OD) or in two distinct routes r and r' (classical or OD), and reconnects the path(s) created using arcs (i, v) and (u, j).
2. Move Node: This operator removes one node i from a route r and inserts it in another r' in the best feasible position. We implemented four variants: classical to classical, classical to OD, OD to OD, OD to classical.
3. 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 best feasible positions. We implemented four variants: classical to classical, classical to OD, OD to OD, OD to classical.
4. Swap Intra-Route: This operator changes the position of two nodes i and j in a given route r, with the respect to the constraints. We implemented two variants: classical and OD.
5. New Route best: This operator initializes a new route r' (i.e., (s,t) for a classical vehicle, (s, v_k) for an OD $k \in K$), only if this choice leads to a less expensive solution. It removes one node i from a given route r and inserts i in a new route r'. We implemented two variants: classical and OD.
6. New Route: This operator initializes a new route r'. It removes one node i from a given route r and inserts i in a new route r'. We implemented one variant: classical.
7. Remove and Insert: This is a particular neighbourhood, which applies sequentially all the four variants of "Move Node" moves.

Shaking. To perturb the current solution we select and then apply two different LS operators. Despite the classical VNS where the shaking phase is completely random, in our approach the selection is a

semi-random choice. Indeed, 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 solution cost, we increment the scores of the shaking moves, otherwise we reduced them. Then, we randomly select the LS operators among those with the best scores.

VND. The *Shaking* phase is followed by a LS phase with the purpose to improve the current solution. We propose a VND described in Algorithm 2.

Algorithm 2: Variable neighborhood descent.

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

4 COMPUTATIONAL STUDY

This section presents the results of our computational experiments. All computations were performed on an Intel Core i7-5700HQ CPU, with 2.60 GHz and 16 GB of RAM. The VNS and the mathematical model were implemented in Java and the model was solved by CPLEX 12.5.

The remainder of this section is organized as follows. We first describe the strategy used to generate the VRPODTWmd instances (Section 4.1), then we present the numerical experiments. We divided the computational study into two parts. The first part aims to assess the performance of the VNS heuristic in terms of both efficiency and effectiveness (Section 4.2). To this end, we consider small-size instances with up to 15 customers and five ODs for which an optimal solution can be obtained by CPLEX. In the second part, the VNS heuristic is tested on large-size instances (Section 4.3).

4.1 Generation of the Instances

We conducted several numerical experiments on different size instances. At first we generated the VRPODTWmd instances based on the classical Solomon VRPTW instances (Solomon, 1987). We created a set of 36 small instances randomly choosing five, 10 and 15 customers and three and five OD destinations. We also created 30 medium instances with 25 and 50 customers, and 15 large instances with 100 customers. To obtain the test instances for the VRPODTWmd, given a VRPTW instance with the customers locations identified by the coordinates (x_i, y_i), we randomly generated the destinations for the ODs, in the square with lower left hand corner $(\min_i\{x_i\}, \min_i\{y_i\})$ and upper right hand corner $(\max_i\{x_i\}, \max_i\{y_i\})$, (see Archetti et. al (Archetti et al., 2016)). We then randomly generated a reasonably time window, considering the planning horizon. In particular, let T_{max} be the maximum duration of the tour, i.e. l_s, and with t^r a random number in the range $[min_{i\in C}(t_{si} + t_{it}), max_{i\in C}(t_{si} + t_{it})]$, where t_{ij} for each $(i, j) \in A$ is the time necessary to travel from i to j. For each OD k, we generated a time window $[e_k, l_k]$, where e_k and l_k are random values in the range $[0, T_{max}]$ and $[(e_k + t_{sk} + t^r), T_{max}]$, respectively.

Figure 1 provides an example of an instance generation.

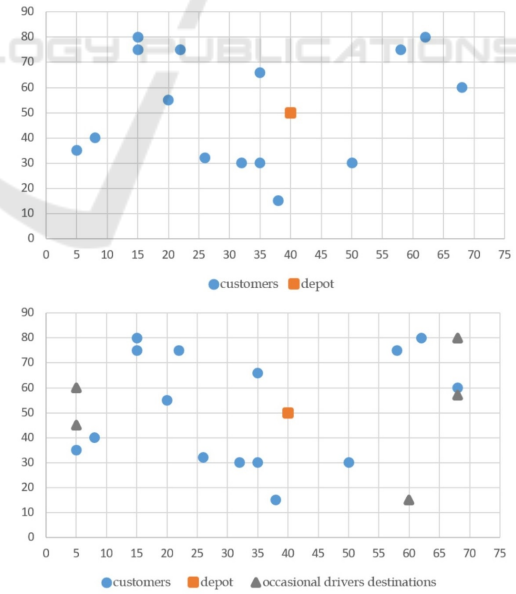

Figure 1: Generation of instance C103C15.

Table 2 reports the characteristics of the instances in terms of number of customers $|C|$, number of classical vehicles P, number of available ODs $|K|$, capacity of the vehicles Q, and capacity of the ODs Q_k.

Table 2: Parameters setting for the generation of the instances.

|C|	P	|K|	Q	Q_k
5	3	3	80	[10,25]
10	3	3	80	[10,30]
15	3	5	80	[15,35]
25	5	10	100	[20,40]
50	8	15	200	[20,40]
100	10	30	400	[20,40]

4.2 Evaluation of the VNS Perfomance

In order to assess the performance of our VNS, we first solved the problem to optimality with CPLEX, we then compared the results with those obtained by the VNS. We set $k_{\max} = 50$ and $improve = 15$.

Tables 3 to 5 compare the solutions obtained for small-size instances. For each table and each algorithm (VNS and resolution with CPLEX), the first column displays the name of the instance, the second one the execution time [ms], the third one is the total cost, the fourth and the fifth columns show the number of classical vehicles (CV) and OD vehicles used, respectively. The two last columns display the optimality gap on cost, calculated as g_{cost}= (Objective$_{VNS}$ − Objective$_{CPLEX}$)/Objective$_{CPLEX}$ and the Speedup (Sp-up), calculated as the ratio between the execution time of CPLEX and that of the heuristic, respectively. The VNS looks very effective, indeed, it finds an optimal solution for the majority of instances. In particular, Table 3 and Table 4 highlight the efficiency of the VNS for instances with five and 10 customers. The proposed algorithm finds the optimal solution for all instances. As shown in Table 5, the VNS is also highly effective for instances with 15 customers, the average gap is 0.3% and it finds the optimal solution for 75% of the instances. Looking at column Sp-up in Tables 3 to 5, it is worth observing that our approach is also more efficient than CPLEX. We can summarize that the VNS finds the optimal solutions for the majority of the instances within short computing times. Our approach clearly outperforms CPLEX.

Table 3: Results for five customers and three occasional drivers instances.

	VNS				OPT					
Test	Time	Cost	CV	OD	Time	Cost	CV	OD	gap	Sp-up
C101C5	99.0	145.4	1	2	60.0	145.4	1	2	0.0%	0.6
C103C5	22.0	111.6	1	2	24.0	111.6	1	2	0.0%	1.1
C206C5	10.0	159.6	1	1	30.0	159.6	1	1	0.0%	3.0
C208C5	5.0	131.9	1	1	38.0	131.9	1	1	0.0%	7.6
R104C5	20.0	107.2	1	2	28.0	107.2	1	2	0.0%	1.4
R105C5	2.0	143.1	2	1	12.0	143.1	2	1	0.0%	6.0
R202C5	23.0	129.5	2	1	46.0	129.5	2	1	0.0%	2.0
R203C5	20.0	170.1	1	1	28.0	170.1	1	1	0.0%	1.4
RC105C5	20.0	143.1	1	2	16.0	143.1	1	2	0.0%	0.8
RC108C5	12.0	164.0	1	2	21.0	164.0	1	2	0.0%	1.8
RC204C5	5.0	107.0	1	2	30.0	107.0	1	2	0.0%	6.0
RC208C5	5.0	124.9	1	2	25.0	124.9	1	2	0.0%	5.0
Avg	20.3	136.5	1.2	1.6	29.8	136.5	1.2	1.6	0.0%	3.1

Table 4: Results for 10 customers and three occasional drivers instances.

	VNS				OPT					
Test	Time	Cost	CV	OD	Time	Cost	CV	OD	gap	Sp-up
C101C10	1028.0	283.2	3	2	127	283.2	3	2	0.0%	0.1
C104C10	16.0	242.4	2	3	315	242.4	2	3	0.0%	19.7
C202C10	36.0	175.4	2	3	51	175.4	2	3	0.0%	1.4
C205C10	16.0	184.7	2	2	97	184.7	1	2	0.0%	6.1
R102C10	19.0	166.4	1	2	68	166.4	1	2	0.0%	3.6
R103C10	286.0	154.0	2	1	316	154.0	2	1	0.0%	1.1
R201C10	99.0	185.2	2	2	73	185.2	3	2	0.0%	0.7
R203C10	17.0	132.9	1	3	100	132.9	1	3	0.0%	5.9
RC102C10	46.0	331.8	2	2	78	331.8	2	2	0.0%	1.7
RC108C10	263.0	330.1	2	2	184	330.1	2	2	0.0%	0.7
RC201C10	28.0	231.1	2	2	34	231.1	2	2	0.0%	1.2
RC205C10	23.0	260.3	2	2	44	260.3	2	2	0.0%	1.9
Average	156.4	223.1	1.9	2.2	123.9	223.1	1.9	2.2	0.0%	3.7

Table 5: Results for 15 customers and five occasional drivers instances.

	VNS				OPT					
Test	Time	Cost	CV	OD	Time	Cost	CV	OD	gap	Sp-up
C103C15	215.0	206.5	1	5	4092.0	206.5	1	5	0.0%	19.0
C106C15	99.0	161.9	2	5	718.0	161.9	2	5	0.0%	7.3
C202C15	85.0	338.3	2	4	2786.0	332.9	2	4	1.6%	32.8
C208C15	10.0	304.7	3	3	351.0	304.7	3	3	0.0%	35.1
R102C15	129.0	297.8	3	4	191.0	297.8	3	3	0.0%	1.5
R105C15	247.0	215.8	2	5	127.0	215.8	1	5	0.0%	0.5
R202C15	30.0	324.4	3	4	5515.0	324.4	3	4	0.0%	183.8
R209C15	782.0	239.4	3	3	3648.0	239.4	3	3	0.0%	4.7
RC103C15	51.0	341.6	3	3	2348.0	341.6	3	3	0.0%	46.0
RC108C15	72.0	252.6	2	5	13848.0	248.6	2	5	1.6%	192.3
RC202C15	144.0	357.3	3	4	390.0	356.3	3	5	0.3%	2.7
RC204C15	23.0	341.1	2	3	831837.0	341.1	2	3	0.0%	36166.8
Average	157.3	281.8	2.4	4.0	72154.3	280.9	2.3	4.0	0.3%	3057.7

4.3 Numerical Results on the Medium-size and Large-size Instances

We now present the computational results on instances with 25, 50 and 100 customers.

with $k_{\max} = 200$ and $improve = 150$.

Tables 6 to 8 summarize the results obtained for this set of instances. For each table, the first column displays the name of the instance, the second one the computational time [ms], the third one the total cost, the fourth and fifth columns show the number of classical and OD routed vehicles, respectively. We also report averages in the last line. Overall, the VNS finds solutions within short computing times. Indeed, the VNS solve instances with 25 nodes in about one second, with 50 nodes in about four seconds, and with 100 nodes in about 20 seconds (about three time less than the time spent by CPLEX to solve instances with 15 customers). Table 6 shows that the solutions generated use, on average, about six ODs out of 10 and about four traditional vehicles. Tables 7 exhibits the same trend. Indeed, the solutions use on average eight out of 15 ODs. The use of ODs becomes more interesting on instances with 100 nodes. Looking at Table 8, it is clear that the majority of the generated solutions use a huge number of ODs.

The advantages of using ODs in transportation

Table 6: Results for instances with 25 customers.

Test	Time	Cost	# traditional	# occasional
C101C25	1684.0	299.2	4	3
C102C25	2563	350.4	5	2
C103C25	2381.0	331.9	3	8
C104C25	716.0	326.8	4	3
C105C25	1028.0	350.8	5	1
R101C25	629.0	326.5	3	9
R102C25	1214.0	293.3	1	10
R103C25	828.0	297.5	2	8
R104C25	754.0	306.7	2	9
R105C25	149.0	288.3	2	9
RC101C25	4234.0	560.7	5	3
RC102C25	2387.0	539.1	5	3
RC103C25	772.0	455.9	4	7
RC104C25	1240.0	455.3	4	6
RC105C25	2058.0	538.1	5	3
Average	1509.1	381.4	3.6	5.6

Table 7: Results for instances with 50 customers.

Test	Time	Cost	# traditional	# occasional
C101C50	1348	521.1	4	7
C102C50	9452	443.4	3	12
C103C50	1355	447.9	3	9
C104C50	19422	437.2	4	7
C105C50	761	461.7	3	9
R101C50	830	727.3	5	15
R102C50	5283	707.4	4	14
R103C50	2801	539.4	3	9
R104C50	1045	509.6	3	11
R105C50	951	662.7	5	6
RC101C50	1247	516.8	3	12
RC102C50	5960	547.8	4	7
RC103C50	17246	561.6	4	7
RC104C50	8229	601.1	4	6
RC105C50	2370	660.9	6	3
Average	5220.0	556.4	3.9	8.9

Table 8: Results for instances with 100 customers.

Test	Time	Cost	# traditional	# occasional
C101	927.0	1344.5	5	28
C102	3031	1376.9	6	24
C103	1036.0	1230.5	6	27
C104	37963.0	1172.1	6	19
C105	1246.0	1431.6	6	27
R101	18538.0	1384.0	9	19
R102	34872.0	1292.9	8	11
R103	23756.0	1157.2	7	13
R104	86965.0	880.6	4	12
R105	6707.0	1337.4	8	11
RC101	23431.0	1425.3	8	27
RC102	25884.0	1488.5	8	9
RC103	30032.0	1165.5	7	10
RC104	11398.0	1085.5	6	17
RC105	769.0	1295.1	9	29
Average	20437.0	1271.2	6.9	18.9

planning have been deeply analyzed in (Macrina et al., 2017), where the effectiveness of delivery configurations that rely on the use of ODs is compared with that obtained by using traditional vehicles only. It has been shown that the use of the ODs may improve the routing plans, generating an interesting cost saving of about 12.5% on small-size instances. On the other hand, general-purpose optimization solvers, like CPLEX, are not able to solve instances with more than 15 customers and five ODs. Thus, it is evident that the availability of a heuristic approach allows to fully exploit the benefits of crowd-shipping also in case of large-size instances. Our computational study confirms that the number of ODs used increases with the increase of the size of the instances. From a practical point of view, this not only means that a cost improvement for the transportation companies is obtained, but also a reduction of the delivery times is determined, with a consequent improvement of the quality of service offered to customers.

5 CONCLUSIONS

We have proposed a variable neighborhood search heuristic for the vehicle routing problem with occasional drivers, time windows, and multiple deliveries. In order to assess the performance of our proposed heuristic, we have solved the problem with CPLEX for small-size instances. We have then conducted a comparative analysis. We have shown that our heuristic is highly performing in terms of effectiveness and efficiency. Overall, the heuristic is less time consuming than CPLEX. We have also shown that it can solve large-size instances within a limited computing time.

ACKNOWLEDGEMENTS

This work was partly supported by MIUR "PRIN2015" funds, project: "Transportation and Logistics in the Era of Big Open Data" - 2015JJLC3E_003 - CUP H52F15000190001.

REFERENCES

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

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

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

Bensinger, G. (2015). Amazon's next delivery drone: You. Wall Street Journal.

Dahle, L., Andersson, H., and Christiansen, M. (2017). The vehicle routing problem with dynamic occasional drivers. In Bektaş, T., Coniglio, S., Martinez-Sykora, A., and Voß, S., editors, *Computational Logistics*, pages 49–63, Cham. Springer International Publishing.

Dahle, L., Andersson, H., Christiansen, M., and Speranza, M. G. (2019). The pickup and delivery problem with time windows and occasional drivers. *Computers and Operations Research*, 109:122–133.

Dayarian, I. and Savelsbergh, M. (2017). Crowdshipping and same-day delivery: employing in-store customers

to deliver online rders. Technical report, optimization-online.

eMarketer (2018). Retail ecommerce sales worldwide, 2017–2021. https://www.emarketer.com/chart/219928/retail-ecommerce-sales-worldwide-2017-2021-trillions-change-of-total-retail-sales.

Gdowska, K., Viana, A., and Pedroso, J. P. (2018). Stochastic last-mile delivery with crowdshipping. *Transportation Research Procedia*, 30:90 – 100.

Landa, R. (2015). *Thinking Creatively in the Digital Age*. Nimble, Blue Ash, Ohio, 1st edition.

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

Macrina, G., Di Puglia Pugliese, L., Guerriero, F., and Laporte, G. (2020). Crowd-shipping with time windows and transshipment nodes. *Computers and Operations Research*, 113.

Macrina, G. and Guerriero, F. (2018). The green vehicle routing problem with occasional drivers. In Daniele, P. and Scrimali, L., editors, *New Trends in Emerging Complex Real Life Problems*. Springer International Publishing, Springer New York LLC.

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