Computers & Industrial Engineering 85 (2015) 120–131

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Variable Neighborhood Search for a Dynamic Rich Vehicle Routing Problem with time windows ☆

Jesica de Armas *, Belén Melián-Batista

Escuela Técnica Superior de Ingeniería Informática, Universidad de La Laguna, 38271 La Laguna, Spain

ARTICLE INFO

Article history:
Received 29 September 2014
Received in revised form 6 February 2015
Accepted 12 March 2015
Available online 21 March 2015

Keywords:
Dynamic Rich Vehicle Routing Problem
Metaheuristics
Variable Neighborhood Search
Degree of dynamism

ABSTRACT

A Dynamic Rich Vehicle Routing Problem with Time Windows has been tackled as a real-world application, in which customers requests can be either known at the beginning of the planning horizon or dynamically revealed over the day. Several real constraints, such as heterogeneous fleet of vehicles, multiple and soft time windows and customers priorities, are taken into consideration. Using exact methods is not a suitable solution for this kind of problems, given the fact that the arrival of a new request has to be followed by a quick re-optimization phase to include it into the solution at hand. Therefore, we have proposed a metaheuristic procedure based on Variable Neighborhood Search to solve this particular problem. The computational experiments reported in this work indicate that the proposed method is feasible to solve this real-world problem and competitive with the best results from the literature. Finally, it is worth mentioning that the software developed in this work has been inserted into the fleet management system of a company in Spain.

1. Introduction

Many practical applications related to logistics in intelligent freight transportation systems lead to vehicle routing problems with varying degrees of difficulty regarding the problem constraints. The basic Vehicle Routing Problem (VRP) is composed of a set of customers that have to be served. A fleet of homogeneous vehicles dispatched from a single depot is used to serve them, returning to the same depot once the routes have been completed. The constraints associated to the problem are that vehicles can carry a maximum capacity and each customer has to be visited once by a single vehicle. Contrary to these classical static vehicle routing problems, real-world applications often include evolution, as introduced by Psaraftis (1980), which takes into consideration the fact that the problem data might change over the planning horizon. Latest developments in fleet management systems and communication technology have enabled people to quickly access and process real-time data. Therefore, Dynamic Vehicle Routing Problems (DVRPs) have been lately given more attention. The aim of DVRPs is to dynamically route customers taking into account not only the requests known at the beginning of the planning horizon, but also new customer requests that arrive over it. Last decade has been characterized by an increasing interest for DVRPs, with solution methods ranging from mathematical programming to metaheuristics. For a survey on DVRPs, we refer the interested reader to the reviews, books, and special issues by Gendreau and Potvin (2004), Ghiani, Guerriero, Laporte, and Musmanno (2003), Ichoua, Gendreau, and Potvin (2007), Larsen, Madsen, and Solomon (2008), and Pillac, Gendreau, Gueret, and Medaglia (2013).

The aim of this work is to solve a real-world DVRP that has been posed to the authors by some companies in the Canary Islands, Spain. The resulting software will be embedded into a fleet management system. The requirements provided by the companies lead to the consideration of several constraints, which have to be integrated into the standard DVRP. In the literature, there is a tremendous number of research papers related to VRP with additional constraints. With the purpose of collecting all these possible constraints, Vidal, Crainic, Gendreau, and Prins (2013) have given the notion of *attributes* of VRPs. Attributes refer to additional constraints that aim to better take into account the specificities of real-world applications. These attributes complement the traditional VRP formulations and lead to a variety of *Multi-Attribute Vehicle Routing Problems* (*MAVRPs*), which are supported by a well developed literature that includes a wide range of heuristics and metaheuristics (Glover, 1986). Furthermore, some MAVRPs combine multiple attributes together, yielding the so-called *Rich VRPs*

☆ This manuscript was processed by Area Editor I-Lin Wang.

* Corresponding author.
E-mail addresses: jdearmas@ull.es (J. de Armas), mbmelian@ull.es (B. Melián-Batista).

http://dx.doi.org/10.1016/j.cie.2015.03.006
0360-8352/

(*RVRPs*) (Schmid, Doerner, & Laporte, 2013). The problem tackled in this work corresponds to this last class of RVRPs. The attributes that are taken into consideration in this work are summarized in the following items.

- *Heterogeneous fleet*. When the number of available vehicles is not limited, the problem is usually referred to as Vehicle Fleet Mix Problem (VFMP). In the case in which the fleet of vehicles is limited, a more difficult version of the problem, called Heterogeneous Fleet VRP (*HFVRP*), is revealed. This work handles a fixed set of heterogeneous vehicles. We refer to the so obtained problem as Fixed HFDVRP (*FHFDVRP*). Most literature papers assume an unlimited number of available vehicles, so that the objective is generally to obtain a solution that either minimizes the number of vehicles and/or total travel cost. However, the real-world problems arising in companies face several resource constraints such as a fixed fleet. Therefore, if there is not any feasible solution for the instance at hand regarding the number of available vehicles, it is required to determine what a good solution would then be for the company (adding more vehicles, letting the drivers work after their working shift, postponing services and maximizing the number of customers served, etc.).
- *Soft and Multiple time windows*. Additional constraints arise if time windows are associated to the depot and customers, obtaining the FHFDVRP with Time Windows (*FHFDVRPTW*). In the implementation carried out in this paper, multiple time windows for customers, which can differ among them, are taken into consideration (Ibaraki et al., 2005, 2008). In any case, each customer is served at maximum once during the planning horizon. Furthermore, the working shifts of the vehicles can be divided into time intervals, which may differ among vehicles. Finally, soft time windows and working shifts are considered, since some of them can be violated. Particularly, if the working shifts corresponding to the vehicles can be extended, extra hours are allowed for the drivers. This leads to additional salary costs; the extra time is more expensive.
- *Customer priority*. The companies under consideration in this work assign priorities to some customers. Depending on these priorities, some services can be postponed until the next day. Together with extending the working shifts of the vehicles, postponing customers services allows the system obtaining valid solutions for the companies. Therefore, in the case in which the fixed fleet of vehicles is not sufficient for serving all customers, allowing extra time and/or postponing customers service are possible alternatives if they are permitted by the companies.
- *Vehicle-Customer restrictions*. There are also vehicle-customer limitations, which indicate that some customers cannot be served by some vehicles. Therefore, there will be a set consisting of vehicle-customer constraints that can be due to several reasons such as road restrictions.

In the rest of the paper, we will refer to the problem considered in this work as Dynamic Rich Vehicle Routing Problem with Time Windows (DRVRPTW).

Additionally to these attributes, different objective functions can be required by the companies to solve the problem at hand. While the optimality criterion of minimizing the total traveled distance is the most commonly used in the VRP literature, more recent approaches use other objective functions. Jozefowiez, Semet, and Talbi (2008) provide an overview of the research into routing problems with several objectives. In addition to the minimization of the total traveled distance, important objectives are the minimization of the number of vehicles in use, the minimization of the total required time and some other objectives related to reach a balance between the routes. In this work, the main objective is to minimize the total traveled distance. Although the problem under consideration in not a multi-objective one, a set of other objective functions are considered together with the main one, as it will be explained below; particularly, minimizing the number of vehicles, extra hours, postponed services and cost. All these functions will be used following a lexicographical order.

Due to the difficulty for solving DRVRPTWs to optimality, heuristics and metaheuristics constitute an increasingly active research area in the literature. In our work, a General Variable Neighborhood Search algorithm (GVNS) (Hansen, Mladenovic, & Moreno Perez, 2010) is proposed. The main differences between the problem tackled in this paper and the ones proposed in the literature are related to the fact that we consider a fixed heterogeneous fleet of vehicles and several real-world constraints/attributes.

The main contributions of this paper rely upon the fact that theDVRPTW including several real-world constraints required by some companies has been tackled. A fixed heterogeneous fleet of vehicles is considered. Moreover, taking into account that the fleet is fixed, there might be customers which cannot be served during the planning horizon and the so obtained infeasibility has to be managed. Two alternative solutions are given in this work; extending the working shifts of the drivers or maximizing the number of customers served postponing the remainder. As far as we know, this is the first work in the literature that uses all the previously explained attributes together in DVRPTWs. Computational experiments over the most common instances in the literature are carried out in this paper. The obtained results are competitive if we compare them with the results in related works. Moreover, some preliminary experiments performed with the fleet management system are quite promising. It is worth mentioning that the static part of the solution method proposed in this work, implemented by means of metaheuristics, has already been integrated into the optimization tool used by the fleet management system of the company (De Armas, Melián-Batista, Moreno-Pérez, & Brito, 2015). Finally, it is important to notice that the algorithm proposed in this work can be run by deactivating any or all the attributes mentioned above. Therefore, it is a general purpose algorithm for solving DVRPTWs.

The rest of the paper is organized as follows. Section 2 reports the related works. Section 3 is devoted to describe the real-world Dynamic Rich Vehicle Routing Problem with Time Windows (DRVRPTW) tackled in this work. Section 4 summarizes the metaheuristic procedure developed to solve the problem at hand. Section 5 reports the computational experiments performed in this work. Finally, the conclusions and future works are given in Section 6.

2. Related work

In general, solution approaches for DVRPTWs can be divided into two main classes: those applied to dynamic and deterministic routing problems without any stochastic information, and those applied to dynamic and stochastic routing problems, in which additional stochastic information regarding the new requests is known. Given the fact that in the real-world application tackled in this paper, the information is dynamically given by a company fleet management system, we will focus on the first class of dynamic problems. In this case, solution methods can be based on either periodic or continuous re-optimization. Periodic optimization approaches firstly generate an initial solution consisting of a set of routes that contain all the static customers. Then, a re-optimization method periodically solves a static routing problem, either when new requests arrive or at fixed time slots (Chen & Xu, 2006). On the other hand, continuous re-optimization approaches carry out the optimization over the day by keeping

high quality solutions in an adaptive memory. In this case, vehicles do not know the next customer to be visited until they finish the service of a request. The following literature references regarding periodic optimization approaches to solve the DVRPTW are worth mentioning. Chen and Xu (2006) proposed a dynamic column generation algorithm for solving the DVRPTW based on their notion of decision epochs over the planning horizon, which indicate the moments of the day when the re-optimization process is executed. Some other papers that make also use of time slices and solve static VRPs are due to Montemanni, Gambardella, Rizzoli, and Donati (2005), Rizzoli, Montemanni, Lucibello, and Gambardella (2007) and Khouadjia, Sarasola, Alba, Jourdan, and Talbi (2012). In these last papers, requests are never urgent and can be postponed since time windows are not handled. On the other hand, the work by Hong (2012) does consider time windows and therefore, some requests can be urgent. Hong proposes a Large Neighborhood Search (LNS) algorithm for real-time vehicle routing problem with time windows, in which each time a new request arrives, it is immediately considered to be included in the current solution. Xu, Wang, and Yang (2013) also takes into account the same problem with urgent requests, but uses a VNS as solving algorithm. Finally, Ghannadpour, Noori, Tavakkoli-Moghaddam, and Ghoseiri (2014) tackle a DVRPTW, in which the time windows are considered as fuzzy. Moreover, they use a homogeneous fleet of vehicles and multiple objective functions.

A tremendous amount of work in the field of vehicle routing problem using Variable Neighborhood Search (VNS) has been published. Bräysy (2003) gives the internal design of the Variable Neighborhood Descent (VND) and Reduced Variable Neighborhood Search (RVNS) algorithms in detail, analyzes the VRPTW problem, and indicates the VND algorithm as one of the most effective ways to solve VRPTW problems. Polacek, Hartl, Doerner, and Reimann (2004) design a VNS to solve the multidepot vehicle routing problem with time windows MDVRPTW. Kytöjoki, Nuortio, Brysy, and Gendreau (2007) design a guided VNS algorithm to handle the 32 existing large-scale VRP problems and compare it with a tabu search algorithm (TS). The result showed that the VNS algorithm was more effective than the TS algorithm in solving time. Goel and Gruhn (2008) introduce the RVNS to solve the general VRP problem including time windows, vehicle constraints, path constraints, travel departure time constraints, capacity constraints, the order models compatibility constraints, multisupplier point of the orders, and transport and service position constraints. Hemmelmayr, Doerner, and Hartl (2009) propose the VNS algorithm for periodical VRP problem. Fleszar, Osman, and Hindi (2008) adopt a VNS algorithm to solve the open-loop VRP problem and test 16 benchmark problems. In summary, several literature papers have proved the effectiveness of developing VNS algorithms to solve a wide variety of VRPs, but none of them has tackled a Dynamic Rich Vehicle Routing Problem with Time Windows.

3. Problem definition

The real-world DRVRPTW solved in this paper is defined by means of a network $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ is the set of nodes, and $\mathcal{A}$ is the set of arcs. It contains the depot, D, and a set of n customer nodes, $\mathcal{C} = (\mathcal{C}_s, \mathcal{C}_d)$, which represent the requests characterized by their demand, location and time windows. $\mathcal{C}_s$ is the set of static customers, that is, those known at the beginning of the planning horizon. On the other hand, $\mathcal{C}_d$ is the set of dynamic customers, which appear over the planning horizon. Customer i can have several time windows during the day, although it is visited at maximum once. The depot has also an associated time window, that is unique, and a set of m heterogeneous vehicles with different capacities. Moreover, associated with each vehicle, k, there are also one or more time intervals of availability during the planning horizon, that represent its working shift and that can be different from one vehicle to another.

Designing a real-time routing algorithm depends to a large extent on how much the problem is dynamic. In order to measure the dynamism of a given problem instance, Lund, Madsen, and Rygaard (1996) defined the *degree of dynamism* of the system as follows:

$$\delta = \frac{|\mathcal{C}_d|}{|\mathcal{C}|} \times 100, \tag{1}$$

where $|\mathcal{C}_d|$ indicates the number of dynamic customers and $|\mathcal{C}|$ the total number of customers. Moreover, since the disclosure time of requests is also important, Larsen (2001) defined the *reaction time* of customer i, that measures the difference between the arrival time, at_i, and the end of the corresponding time window, b_{iw_i}. Notice that longer reaction times indicate that there is more flexibility to insert any new request into the existing routes. Therefore, the effective degree of dynamism provided by Larsen is stated as follows:

$$\delta^e_{TW} = \frac{1}{N}\sum_{i \in \mathcal{C}} \left(1 - \frac{b_{iw_i} - at_i}{T}\right), \tag{2}$$

where T is the length of the planning horizon.

In order to solve the DRVRPTW, firstly, we need to obtain the initial routes containing the static customers $\mathcal{C}_s$. This means that we must start from a solution to RVRPTW, which consists in determining a set of routes of minimal total traveled distance, starting and ending at a depot, such that every customer in $\mathcal{C}_s$ should be visited exactly once by one vehicle. The sum of the demands associated with the nodes contained in a route never exceeds the corresponding vehicle capacity. Moreover, each node must be visited within its time window and it should be done within the corresponding working shift of the used vehicle. Notice that, given the fact that the fleet of vehicles of the company is fixed, it might not be possible to obtain a feasible initial solution. However, in order to obtain an initial solution valid for the company, it is possible to allow exceeding the working shifts of the vehicles, using extra time, which incurs additional cost. Furthermore, we might allow postponing customer services, what means that some customers might not be served during the current planning horizon. If these two conditions were not permitted, every node that cannot be inserted in a plan satisfying the problem constraints, has to be rejected. Nevertheless, in order to provide the company with a solution that includes all customer requests, the constraints can be relaxed so that the time windows of the customers can be exceeded. It means that infeasibilities appear.

Therefore, although minimizing the total traveled distance is the main objective when determining the set of routes, additional objective functions that have to be minimized using a hierarchical approach, have been taken into account. Hierarchical evaluation means that the objective functions are considered in a certain lexicographical order, so that if two selected solutions have equal values for an objective function, then the next one in the order is considered to break ties. The lexicographical order considered in the solution of the problem is the following.

- Total infeasibility (Sum of the times that the customers time windows are exceeded).
- Number of postponed services.
- Number of extra hours (Sum of the times that the vehicles working shifts are exceeded).
- Total traveled distance.
- Total number of routes.

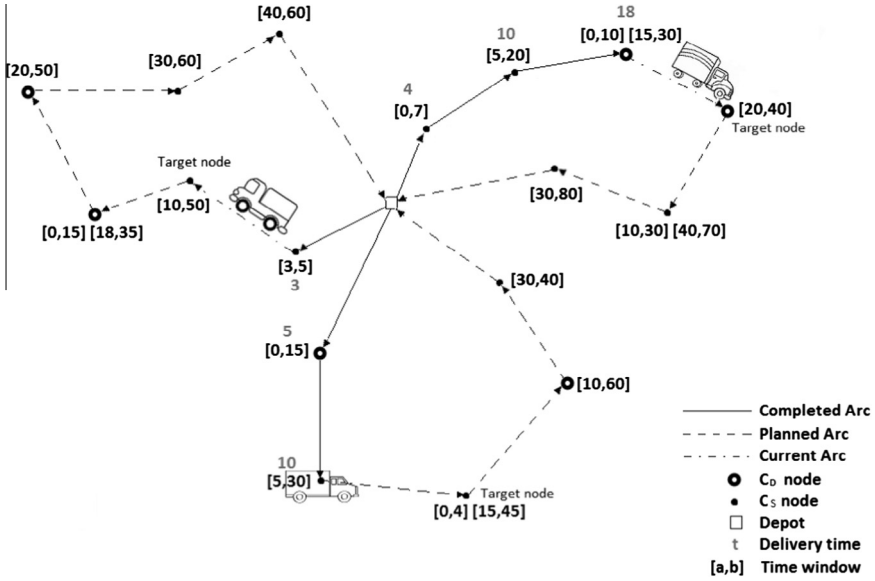

Fig. 1. Solution example.

- Time balance (difference between the largest and shortest routes made by one vehicle regarding time) (Melian-Batista, De Santiago, AngelBello, & Alvarez, 2014).
- Salary costs for each driver of each vehicle (extra hours are more expensive).

Note that this order for evaluating the objective functions intends to firstly eliminate infeasibility, number of postponed services and number of extra hours required by the vehicles. If we are able to reduce to zero these three objective function values, solutions that adjust to the company resources are obtained. Therefore, in order to obtain the best possible solution, it is given higher priority to those solutions that do not have infeasibilities, include all the services and accomplish the working shifts of the drivers.

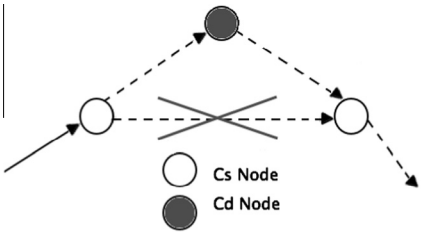

Fig. 2. Insertion of a dynamic customer.

Once the initial static RVRPTW is solved, we have an initial plan composed of a number of routes, so that the routes can be started and the dynamic customers can appear within the planning horizon. The DRVRPTW is strongly related to the static RVRPTW, as it can be described as a routing problem in which information about the problem can change during the optimization process. As conventional static RVRPTWs are NP-hard, DRVRPTW also belongs to the class of NP-hard problems. It is a discrete-time dynamic problem, and can be viewed as a series of instances; each instance is a static problem, which starts at a certain time and must be solved within a specific deadline. Fig. 1 shows a solution example, in which the static and dynamic nodes can be distinguished. The solid lines are already traversed by the vehicles, whereas dashed lines can be modified in order to insert a new dynamic customer, as

Table 1
Computational results of type 1 problems for various degrees of dynamism.

Type	Dod (%)	Avg. vehicle number			Avg. total distance			Avg. insertion time			Ratio post. customers		
		GVNS	Ref.	ARE (%)	GVNS	Ref.	ARE (%)	GVNS	Ref.	ARE (%)	GVNS (%)	Ref. (%)	ARE (%)
R1	90	14.67	14.25	2.95	1250.38	1335.94	−6.84	14.50	17.43	−20.21	3.83	2.33	64.38
	70	14.75	14.33	2.93	1267.78	1331.34	−5.01	10.95	21.73	−98.45	3.08	1.75	76.00
	50	14.58	14.08	3.55	1267.47	1295.81	−2.08	11.84	28.27	−138.77	1.92	0.67	186.57
	30	14.25	13.92	2.37	1256.04	1286.63	−2.44	15.70	46.99	−199.30	1.58	0.58	172.41
	10	14.17	13.50	4.96	1250.16	1257.08	−0.55	15.29	67.99	−344.67	0.50	0.17	194.12
C1	90	10.67	10.78	−1.03	963.33	1039.77	−7.93	7.81	6.60	18.33	0.00	0.22	−100.00
	70	11.33	10.78	5.10	1009.47	1031.68	−2.20	7.67	10.79	−40.68	0.00	0.22	−100.00
	50	11.00	10.89	1.01	992.97	1001.18	−0.83	6.22	19.01	−205.63	0.00	0.22	−100.00
	30	11.56	10.56	9.47	949.95	962.08	−1.28	9.13	28.02	−206.90	0.00	0.33	−100.00
	10	10.56	10.56	0.00	898.30	895.77	0.29	13.74	45.40	−230.42	0.00	0.22	−100.00
RC1	90	14.63	14.00	4.50	1470.45	1513.94	−2.45	15.39	17.31	−12.48	1.88	2.00	−6.00
	70	14.88	13.88	7.20	1489.28	1511.29	−1.48	13.43	25.32	−88.53	2.13	1.88	13.30
	50	14.50	13.63	6.38	1484.01	1514.72	−2.07	13.72	48.78	−255.54	1.75	1.38	26.81
	30	14.38	13.88	3.60	1471.00	1492.22	−1.29	16.51	45.26	−174.14	1.00	1.13	−11.50
	10	13.50	13.38	0.90	1417.07	1436.23	−1.04	23.01	83.52	−262.97	0.50	1.13	−55.75
Avg		13.30	12.83	3.59	1229.18	1260.38	−2.56	12.99	34.16	−150.69	1.21	0.95	10.69

Table 2
Computational results of type 2 problems for various degrees of dynamism.

Type	Dod (%)	Avg. vehicle number			Avg. total distance			Avg. insertion time			Ratio post. customers		
		GVNS	Ref.	ARE (%)	GVNS	Ref.	ARE (%)	GVNS	Ref.	ARE (%)	GVNS (%)	Ref. (%)	ARE (%)
R2	90	4.00	3.55	12.68	1086.78	1047.82	3.72	16.47	13.20	24.77	0.00	0.09	−100.00
	70	4.36	3.64	19.78	1078.03	1032.04	4.46	12.74	20.15	−36.77	0.00	0.09	−100.00
	50	4.55	3.82	19.11	1071.83	1016.52	5.44	11.96	30.03	−60.17	0.00	0.00	0.00
	30	4.73	4.91	−3.67	1035.60	985.49	5.08	10.18	57.07	−82.16	0.00	0.00	0.00
	10	5.27	6.36	−17.14	1000.90	950.00	5.36	9.48	68.58	−86.18	0.00	0.09	−100.00
C2	90	3.38	3.25	4.00	668.99	636.79	5.06	16.67	6.12	172.39	0.00	0.00	0.00
	70	3.38	3.13	7.99	672.95	636.47	5.73	14.03	10.01	40.16	0.00	0.00	0.00
	50	3.13	3.13	0.00	623.10	604.98	3.00	20.25	16.80	20.54	0.00	0.00	0.00
	30	3.25	3.63	−10.47	624.81	651.42	−4.08	34.82	29.87	16.57	0.00	0.00	0.00
	10	3.25	3.00	8.33	615.93	594.67	3.58	80.78	59.70	35.31	0.00	0.00	0.00
RC2	90	4.63	4.00	15.75	1275.93	1257.19	1.49	28.05	11.34	147.35	0.00	0.13	−100.00
	70	5.13	3.88	32.22	1234.36	1239.46	−0.41	16.07	19.26	−16.56	0.00	0.00	0.00
	50	5.88	4.25	38.35	1200.26	1190.54	0.82	11.46	27.84	−58.84	0.00	0.13	−100.00
	30	5.88	5.38	9.29	1172.33	1166.04	0.54	11.68	41.51	−71.86	0.00	0.25	−100.00
	10	6.13	6.75	−9.19	1153.43	1103.30	4.54	13.27	55.55	−76.11	0.00	0.00	0.00
Avg		4.46	4.18	8.47	967.68	940.85	2.95	20.53	31.14	−2.10	0.00	0.05	−40.00

shown in Fig. 2. Note that a customer can have more than one time window, although it has to be visited at maximum once during the day.

The fundamentals kept to solve the DRVRPTW are the following:

- Vehicles assigned to routes must serve the planned static customers, and vehicles in the depot are dispatched according to the actual needs. If a dynamic customer cannot be inserted into the current routes and there are unused vehicles, a new route can be created to serve the customer and the departure time of the vehicle will be the sum of the arrival time of the customer and the processing time of the insertion. This processing time corresponds to the computational time needed to decide where to insert the new customer in the plan of routes.
- Dynamic customer requests are received in real time, and serviceability of requesting customer is immediately verified. The corresponding customer is acceded to the planning routes as quickly as possible.
- A customer is called target customer of a vehicle if the vehicle is moving towards it or is serving its predecessor customer, and must be carefully adjusted. Adjusting a target customer, i.e., inserting a new customer before it, will change the line in which a vehicle is moving, and can cause confusion in the traveling line. For this reason, adjusting target customers should be avoided as much as possible, and, in this work, it will be only permitted in the two following cases:
  - The time window of the new customer would be violated otherwise.
  - Adjusting the target customer avoids some constraints violations.
- The time spent on doing the insertion of a new customer is taken into account, because meanwhile vehicles continue moving.

4. Solution approach

Our interest is focused on metaheuristic methodologies for solving the DRVRPTW that are capable of producing applicable high quality solutions within reasonable computing times. The solution method proposed in this work to solve the DRVRPTW is summarized in Algorithm 1. First of all, an initial solution consisting of all the static customers is generated by using an adapted Solomon Heuristic (Solomon, 1987). The obtained solution is then improved running the General Variable Neighborhood Search (GVNS) (Hansen et al., 2010) described in Algorithm 2, developed in De Armas et al. (2015) to solve the static problem with all the constraints required by the companies for which this dynamic problem has also to be solved. This process (lines 3–4) is iterated for a certain number of iterations and the best reached solution is selected to be implemented by the company. In this step, all the requests known at the beginning of the planning horizon are already inserted in a route. Then, the dynamic heuristic is applied for each new customer that appears while the vehicles are working.

In the next sections, each of these steps are described in detail, from the creation of the initial solution to obtaining dynamic solutions.

Algorithm 1. General algorithm

```
// Create solution S* containing all static customers
1  Initialize solution S*;
2  while (a maximum number of iterations N is not reached) do
3  |  S ← Run Solomon Heuristic;
4  |  S' ← Apply GVNS to S;
5  |  if (S' is better than S*) then
6  |  |  S* ← S';
// Insert dynamic customers at their arrival times
7  while (a new dynamic customer, i, appears) do
8  |  if (i must be adjusted because of its time window) then
9  |  |  Insert customer i in a route before a target customer;
10 |  |  Apply GVNS;
11 |  |  Continue;
12 |  else
13 |  |  Find the closest feasible route r to insert i without any accumulated infeasibility;
14 |  |  Find the best position p in r where the evaluation of objective functions are better;
15 |  |  if (r and p exists) then
16 |  |  |  Insert customer i in position p inside r;
17 |  |  |  Apply GVNS;
18 |  |  |  Continue;
   |  |  ...
```

```
while (a new dynamic customer, i, appears (continuation)) do
...
19  if (i can be adjusted to eliminate constraints violations) then
20      Insert customer i in a route before a target customer;
21      Apply GVNS;
22      Continue;
23  else if (i can be inserted in a new route r) then
24      Insert customer i in route r;
25      Apply GVNS;
26      Continue;
27  if (extra time is permitted) then
28      Find the closest feasible route r to insert i without any
        accumulated infeasibility, permitting extra time;
29      Find the best position p in r where the evaluation of
        objective functions are better;
30      if (r and p exists) then
31          Insert customer i in position p inside r;
32          Apply GVNS;
33          Continue;
34  if (extra time is permitted & i can be inserted in a new
        route r) then
35      Insert customer i in route r using the vehicle that
        needs the smallest extra time;
36      Apply GVNS;
37      Continue;
38  if (the priority of customer i < minPriority) then
39      Postpone customer i until the next day;
40      Apply GVNS;
41      Continue;
42  if (there is an alternative permitted customer j in a route
        r without any accumulated infeasibility, that let the
        insertion of i) then
43      Postpone customer j;
44      Insert customer i in route r;
45      Apply GVNS;
46      Continue;
47  if (infeasibilities are permitted) then
48      Insert customer i in the route that supposes the
        smallest infeasibility (if it coincides among routes, consider
        the remaining objective functions)
49      Apply GVNS;
50 Report infeasibility to the company;
```

4.1. Creation of the initial solution with static customers $\mathcal{C}_s$

In order to generate an initial solution, an ordering of the available vehicles is obtained according to which the vehicles are selected to create the routes. This ordering is given taking into account the capacity of each vehicle, in such a way that vehicles with greater capacity are selected before. If there are multiple vehicles with the same capacity, then they will be sorted according to the number of consecutive hours that the vehicle is available, so that vehicles having larger working shifts are selected first. Once having the selection order of the vehicles, the routes are created one after the other. To create a route, a vehicle and a seed customer, which will be selected among the customers that are the farthest from the depot, have to be chosen. Each customer is then attempted to be inserted, but if it is not compatible with the vehicle due to restrictions, the next vehicle in the sorted list is chosen. After inserting the seed customer, the proposed procedure follows the Solomon algorithm (Solomon, 1987), establishing the route locations where to insert each unplanned customer and selecting the best customer to be inserted. When no more customers can be inserted into the current route, the next one is started to be created.

4.2. General Variable Neighborhood Search

General Variable Neighborhood Search (GVNS) is a metaheuristic for solving combinatorial and global optimization problems based on a simple principle; systematic changes of neighborhoods within the search. Many extensions have been made, mainly to be able to solve large problem instances (Hoeller, Melian, & Voss, 2008; Melian, 2006; Moreno-Vega & Melian, 2008).

GVNS metaheuristic starts from an initial solution. Then, a so-called shaking step is performed by randomly selecting a solution from the first neighborhood. This is followed by applying an iterative improvement algorithm. This procedure is repeated as long as a new incumbent solution is found. If not, one switches to the next neighborhood (which is typically larger) and performs a shaking step followed by the iterative improvement. If a new incumbent solution is found, one starts with the first neighborhood; otherwise one proceeds with the next neighborhood, and so forth. Starting from an initial solution, this metaheuristic consists of the shaking, the local search, and the move decision phases, which are explained below.

Algorithm 2 shows this process, where $\mathcal{N}_k(k=1,\ldots,k_{max})$ is a finite set of neighborhood structures, and $\mathcal{N}_k(s)$ the set of solutions in the kth neighborhood of a solution s. Usually, a series of nested neighborhoods is obtained from a single neighborhood by taking $\mathcal{N}_1(s)=\mathcal{N}(s)$ and $\mathcal{N}_{k+1}(s)=\mathcal{N}(\mathcal{N}_k(s))$, for every solution s. This means that a move to the k-th neighborhood is performed by repeating k times a move into the original neighborhood. A solution $s'\in S$ is a *local minimum* with respect to $\mathcal{N}_k$ if there is no solution $s\in\mathcal{N}_k(s')\subseteq S$ better than s' (i.e., such that $f(s)<f(s')$ where f is the objective function of the problem). Moreover, $\mathcal{N}_l$, $(l=1,\ldots,l_{max})$ is the finite set of neighborhood structures that will be used in the local search.

Therefore, in this algorithm we can see that the loop corresponding to lines 2–11 is performed for a number of iterations, M, set by the computational experience. The processes of shaking, local search and move decision in lines 5, 6 and 11, respectively, are iteratively performed until $k=k_{max}$. In

Algorithm 2. General Variable Neighborhood Search (GVNS)

```
1  Generate an initial solution s.
// Iterations.
2  while (the stopping condition is not met (M is not
       reached))do
3      (1) Set k ← 1;
4      (2) Repeat the following steps until k = k_max:
5          (a) Shaking. Generate a point s' at random from the k^th
           neighborhood of s (s' ∈ N_k(s)).
6          (b)Local search by VND.
7              (b1) Set l ← 1;
8              (b2) Repeat the following steps until l = l_max:
9                  – Exploration of neighborhood. Find the best
               neighbor s of s in N_l(s);
10                 – Move or not. If f(s'') < f(s'), set s' ← s'' and l ← 1;
               otherwise, set l ← l + 1;
11         (c) Move or not. If this local optimum is better than the
           incumbent, move there (s ← s''), and continue the search with N_1
           (k ← 1); otherwise, set k ← k + 1.
```

the first place, the shaking step generates a solution s' at random from the kth neighborhood of s ($s' \in \mathcal{N}_k(s)$). Then, a local search based on *VND* is performed from s' to obtain a solution s''. The VND procedure uses the $\mathcal{N}_l$ neighborhoods.

4.2.1. Shaking

Shaking is a key process in the GVNS algorithm design. The main purpose of the shaking process is to extend the current solution search space, to reduce the possibility that the algorithm falls into the local optimal solution in the follow-solving process, and to get the better solution. The set of neighborhood structures used for shaking is the core of the GVNS.

The set of neighborhoods selected for the shaking process of the GVNS are not nested, and different kinds of movements are implemented following the ideas described by Repoussis, Paraskevopoulos, Tarantilis, and Ioannou (2006). The proposed sequence of movements is defined as follows: *GENI*, *Or-opt*, *CROSS*, 2-*opt*, *relocate* and *swapInter*. This sequential selection is applied based on cardinality, which implies moving from relatively poor to richer neighborhood structures. The *GENI* operator (Gendreau, Hertz, & Laporte, 1992) chooses a customer from a route and inserts it into other route between the two closest customers to the previous one. The *Or-opt* operator (Or, 1976) relocates a chain of two consecutive customers of a route. The *CROSS* operator (Taillard, Badeau, Gendreau, Guertin, & Potvin, 1997) selects a subsequence of two customers from a route, other subsequence of two customers from other route, and interchange both subsequences. The 2-*opt* operator (Croes, 1958) chooses two customers of a route and invert the sequence of customer visited between them. The *relocate* operator (Cassani & Righini, 2004) deletes a customer from a route and insert it into another route. The *swapInter* operator selects a customer from a route, other customer from other route, and swaps them. Note that these movements are performed in the case in which no more infeasibility, postponed services or extra hours are incurred.

4.2.2. Local search

In a GVNS algorithm, local search procedures will search the neighborhood of a new solution space obtained through shaking in order to achieve a locally optimal solution. Local search is the most time-consuming part in the entire GVNS algorithm framework and decides the final solution quality, so that computational efficiency must be considered in the design process of local search algorithm.

As explained before, $\mathcal{N}_l, (l = 1, \ldots, l_{max})$ is the finite set of neighborhood structures that will be used in the local search, conducted by a Variable Neighborhood Descent (VND). The Variable Neighborhood Descent (VND) method is obtained if the change of neighborhoods is performed in a deterministic way. Its steps are presented in Algorithm 3. The sequence of movements considered in this work is the following: relocate, swapIntra and swapInter. The two first ones work as explained above, and the *swapIntra* operator chooses two customers from a route and swaps them.

Algorithm 3. Variable Neighborhood Descent (VND)

Function VND (s, l_{max}).
while (improvement is obtained)**do**
  Set $l \leftarrow 1$;
  while ($l <= l_{max}$) **do**
    $s' \leftarrow argmin_{y \in \mathcal{N}_l(s)} f(s)$
    NeighborhoodChange(s, s', l);//Change neighborhood

4.2.3. Move decision

The last part of the heuristic concerns the acceptance criterion. Here we have to decide whether the solution produced by GVNS will be accepted as a starting solution for the next iteration. The hierarchical evaluation of objectives explained in Section 3 has been used for this purpose.

4.3. Insertion of dynamic customers

Once the selected initial solution is being implemented, new dynamic customers might be revealed over the planning horizon, which have to be inserted in any route. As explained above, a customer is called target customer of a vehicle if the vehicle is moving towards it or is serving its predecessor customer. In this work, adjusting a target node, i.e., inserting a dynamic customer before the target node, will be only permitted in two cases: when the time window of the dynamic customer would be violated otherwise, and when adjusting a target node avoids some constraints violations.

Therefore, let us suppose that the dynamic customer i arrives at time at_i. As reported in line 8, Algorithm 1 first tries to adjust customer i if its time window will be violated otherwise. In this case, the customer is inserted before a target customer and the GVNS algorithm is applied in order to improve the solution. It is necessary to clarify that every time the GVNS algorithm is used henceforth, its operators are applied only after the target customer in each route, since the other customers have been already visited and their order cannot be changed.

If the customer does not need to be adjusted, then a feasible existing route where to insert the new customer is searched (lines 13 and 14). For each of these routes, from the last visited customer to the last customer in the route, it is searched the feasible insertion point with the best result for the evaluation of the objectives. If it is found, the customer is inserted and the GVNS algorithm is applied in order to improve the resulting solution.

If inserting the customer has not been possible, then we will try to adjust the customer to avoid constraints violations (line 19). In this case, the customer is inserted before a target customer and the GVNS algorithm is applied in order to improve the solution. In other case, the new customer is inserted in a new route and the GVNS algorithm is applied, if there is an available vehicle which can serve it.

At this point, if the new customer has not been inserted yet, then we try to use extra time if permitted (lines 27–37). If the customer cannot still be inserted, we try to postpone its service if the minimum priority permits to do this (line 38). Otherwise, we try to postpone the services of other customers that have not been visited yet and allow to insert the new one in the route, if the minimum priority permits to do this (line 42).

Finally, if the new customer has not been inserted or its service postponed yet, it will be inserted into the route that involves less infeasibility (line 47), also taking into account the evaluation of the objectives.

This way, we can manage two possibilities. The first one is rejecting the customers that cannot be feasibly inserted, as it is done in related works from the literature (Hong, 2012; Xu et al., 2013). This can be made by setting the minimum priority to a high value. The second possibility is permitting time windows infeasibilities in the customers in order to provide a solution with all the customers. To the best of our knowledge, it has not been proposed in the related literature.

At this point, it is worth mentioning that this process of deciding where to introduce a new dynamic customer in the planification of routes, takes a certain time that must be taken into

account in real applications. Otherwise, vehicles can have changed their position when the planification is modified.

5. Experimental results

This section is devoted to thoroughly describe the computational experiments carried out in this work to assess the quality of the solutions provided by the algorithm developed to solve a real-world DRVRPTW. The test work has been done using a computer with Intel(R) Core(TM) i5-2320 CPU, 3.00 GHz, 6 Gb RAM and a Linux operating system. The algorithm has been coded in C++.

As mentioned before, the requirements provided by the real companies lead to the consideration of several constraints, which have to be integrated into the standard problem. For example, the use of a fixed set of heterogeneous vehicles, customers with priorities and several time windows, or vehicle–customer restrictions. Solving this problem with optimality is really important for companies, which obtain a cost saving through it.

Even though our algorithm has been developed to solve a real-world DRVRPTW, which has many different attributes and constraints, it would be advisable to know if the results obtained using the standard instances in the literature, which do not take into account these attributes, are competitive with the best ones in related works. Therefore, in order to compare our results with the results in the literature, we have used the standard VRPTW Solomon benchmark instances.[1] In these benchmark problems, 100 nodes are distributed in an Euclidean plane of 100×100 square, and the travel times between nodes are equal to the corresponding Euclidean distances. There are six types of problems, named $R1, R2, C1, C2, RC1$ and $RC2$, each with 8–12 problems. Specifically, the data set designed by Lackner (2004) is adopted for dynamic tests.[2] Each problem has five groups of dynamic data which are used to depict five different degrees of dynamism of $90\%, 70\%, 50\%, 30\%$ and 10%, respectively. Furthermore, our algorithm has been tested with real instances, which present the whole set of real constraints. Starting from four real instances of a company in Canary Islands, $n1, n2, n3$, and $n4$, dynamic instances have been generated[3] for the five different degrades of dynamism, $90, 70, 50, 30$ and 10. Random reaction times have been used for this purpose.

Three kinds of experiments have been carried out. For the first one, we have solved the test problem instances avoiding extra hours for vehicles and time window infeasibilities for customers, and allowing postponed customer services. This experiment has been performed in order to obtain the most comparable results with the results in the literature. For the second experiment, we have solved the test problem instances avoiding extra hours for vehicles and postponed customer services, and allowing time window infeasibilities for customers. In this case, we have provided solutions which include all customer requests relaxing time windows for customers. Third experiment analyzes the results obtained over real instances.

The recent works in the related literature (Hong, 2012; Xu et al., 2013), take into account similar considerations to those of our work regarding the basic dynamic problem. The main differences of the problem tackled in this paper and the ones proposed by Hong (2012) and Xu et al. (2013) are on one hand, the fact that we consider a heterogeneous fleet of vehicles, and on the other hand, the fact that we consider many real constraints. For these reasons, we have taken the results in these works as reference to assess the quality of our method.

[1] http://web.cba.neu.edu/msolomon/problems.htm.
[2] http://www.fernuniversitaet-hagen.de/WINF/inhalte/benchmark_data.htm.
[3] https://sites.google.com/site/gciports/vrptw/dynamic-vrptw.

In both articles from the literature, the average time spent in doing the insertion of a new dynamic customer is about 30 s. This delay may lead to significant changes in the routes state, so that we take it into account. The Solomon's test instances use an unspecified unit of time to define the service times and time windows. However, it does not make sense considering that the units are seconds, overall in relation to the time spent doing customer insertions. The next significative time unit is the minute. For this reason, we have considered the minute as the smallest possible time unit, and therefore we have fixed the delay to 0.5 units (30 s = 0.5 min). None of the works taken as reference make a real discussion about this question.

The parameters M and N used in the GVNS algorithm have been statistically set to 20 and 10, respectively.

5.1. Comparison with the literature for the DVRPTW

For the first experiment carried out in this work, we have run 15 executions for each single Solomon's test instance under different degrees of dynamism, and we have chosen the best results among them to calculate the average values of each group of instances ($R1, R2, C1, C2, RC1$ and $RC2$). Then, we have selected the best results for each group of instances between the ones given in Hong (2012) and Xu et al. (2013). In order to make a valid comparison, the extra time has not been permitted in the executions, but postponing customers has been enabled. In this way, Tables 1 and 2 give the comparison results. For both tables, first column shows the set of test problem instances and the second column shows degrees of dynamism. The next columns show the average number of vehicles, the average total distance, and the average insertion time (in seconds) obtained using our method based on General Variable Neighborhood Search (*GVNS*) and the one in work Hong (2012) (*Ref.*). We also calculate the relative error (*ARE*(%)). Finally, the last two columns present the ratio of postponed customers using our method and the one in Hong (2012).

As can be seen in the average row of Table 1, although we improve the number of vehicles only a few times, in most cases, our distance results improve the literature ones. It is remarkable that in Table 1, for the group of instances $C1$, the distances obtained by GVNS are lower than the ones obtained in Hong (2012) despite that GVNS does not postpone any customer. Moreover, for the degrees of dynamism 90, 30 and 10 of instances $RC1$, we also obtain lower distances with a shorter number of postponed services. In general, there are many cases in which our ratio of postponed customers is lower. This means that we are including more customers into the solution and even in those cases our total distances are lower.

For test problem instances in Table 2, we never postpone any customers. In addition, our insertion times are substantially lower than the best ones, and this is really important because companies need a customer insertion process as quick as possible in order to obtain an efficient system.

If instead of selecting the best results among the 15 executions to calculate the average values of each group of instances, we calculate the average of the 15 executions, we obtain Tables 3 and 4. Although in this case the ratio of postposed customers is better, obviously, in general results have less quality than before. Notice that the works taken as references Hong (2012) and Xu et al. (2013) do not calculate these averages.

5.2. Computational results for the DRVRPTW

In order to provide a solution which includes all customer requests, the constraints can be relaxed so that time windows of customers are permitted to be exceeded, such as indicated by the companies. As mentioned above, it means that infeasibilities

Table 3
Computational results of type 1 problems for various degrees of dynamism using averages.

Type	Dod (%)	Avg. vehicle number			Avg. total distance			Avg. insertion time			Ratio post. customers		
		GVNS	Ref.	ARE (%)	GVNS	Ref.	ARE (%)	GVNS	Ref.	ARE (%)	GVNS (%)	Ref. (%)	ARE (%)
R1	90	15.34	14.25	7.68	1328.74	1335.94	−0.54	15.89	17.43	−8.84	3.33	2.33	43.06
	70	15.31	14.33	6.85	1340.38	1331.34	0.68	12.45	21.73	−42.71	2.42	1.75	38.10
	50	15.33	14.08	8.86	1340.17	1295.81	3.42	12.32	28.27	−56.42	1.67	0.67	148.76
	30	14.96	13.92	7.44	1312.73	1286.63	2.03	17.21	46.99	−63.38	1.17	0.58	101.15
	10	14.73	13.5	9.14	1296.59	1257.08	3.14	16.69	67.99	−75.45	0.33	0.17	96.08
C1	90	11.37	10.78	5.48	1092.18	1039.77	5.04	8.94	6.60	35.45	0.00	0.22	−100.00
	70	11.92	10.78	10.56	1150.27	1031.68	11.49	8.51	10.79	−21.13	0.00	0.22	−100.00
	50	11.81	10.89	8.49	1129.58	1001.18	12.83	7.32	19.01	−61.49	0.00	0.22	−100.00
	30	11.81	10.56	11.88	1081.52	962.08	12.41	10.17	28.02	−63.70	0.00	0.33	−100.00
	10	11.36	10.56	7.53	986.99	895.77	10.18	14.87	45.40	−67.25	0.00	0.22	−100.00
RC1	90	15.62	14.00	11.55	1587.89	1513.94	4.88	15.89	17.31	−8.20	1.50	2.00	−25.00
	70	15.88	13.88	14.43	1614.43	1511.29	6.82	14.72	25.32	−41.86	1.25	1.88	−33.51
	50	15.51	13.63	13.78	1579.34	1514.72	4.27	14.05	48.78	−71.20	0.88	1.38	−36.59
	30	15.22	13.88	9.63	1551.93	1492.22	4.00	16.89	45.26	−62.68	0.63	1.13	−44.69
	10	14.25	13.38	6.50	1474.09	1436.23	2.64	24.52	83.52	−70.64	0.25	1.13	−77.88
Avg		14.03	12.83	9.32	1324.46	1260.38	5.55	14.03	34.16	−45.30	0.89	0.95	−19.37

Table 4
Computational results of type 2 problems for various degrees of dynamism using averages.

Type	Dod (%)	Avg. vehicle number			Avg. total distance			Avg. insertion time			Ratio post. customers		
		GVNS	Ref.	ARE (%)	GVNS	Ref.	ARE (%)	GVNS	Ref.	ARE (%)	GVNS (%)	Ref. (%)	ARE (%)
R2	90	3.88	3.55	9.26	1181.31	1047.82	12.74	17.05	13.20	29.17	0.00	0.09	−100.00
	70	4.22	3.64	16.05	1161.98	1032.04	12.59	13.41	20.15	−33.45	0.00	0.09	−100.00
	50	4.49	3.82	17.56	1153.79	1016.52	13.50	12.58	30.03	−58.11	0.00	0.00	0.00
	30	4.77	4.91	−2.86	1112.92	985.49	12.93	10.86	57.07	−80.97	0.00	0.00	0.00
	10	5.49	6.36	−13.66	1054.82	950.00	11.03	10.75	68.58	−84.32	0.00	0.09	−100.00
C2	90	3.66	3.25	12.56	749.33	636.79	17.67	17.85	6.12	191.67	0.00	0.00	0.00
	70	3.71	3.13	18.48	722.45	636.47	13.51	14.91	10.01	48.95	0.00	0.00	0.00
	50	3.53	3.13	12.62	670.23	604.98	10.79	22.34	16.80	32.98	0.00	0.00	0.00
	30	3.36	3.63	−7.48	670.88	651.42	2.99	35.92	29.87	20.25	0.00	0.00	0.00
	10	3.53	3.00	17.78	660.93	594.67	11.14	85.73	59.70	43.60	0.00	0.00	0.00
RC2	90	8.02	4.00	100.42	2032.46	1257.19	61.67	29.51	11.34	160.23	0.00	0.13	−100.00
	70	4.94	3.88	27.36	1359.18	1239.46	9.66	17.18	19.26	−10.80	0.00	0.00	0.00
	50	5.39	4.25	26.86	1311.97	1190.54	10.20	12.86	27.84	−53.81	0.00	0.13	−100.00
	30	5.83	5.38	8.43	1278.62	1166.04	9.65	13.54	41.51	−67.38	0.00	0.25	−100.00
	10	6.01	6.75	−10.99	1240.55	1103.30	12.44	13.98	55.55	−74.83	0.00	0.00	0.00
Avg		4.72	4.18	15.49	1090.76	940.85	14.83	21.90	31.14	4.21	0.00	0.05	−40.00

Table 5
Computational results of type 1 problems for various degrees of dynamism and infeasibilities.

Type	Dod (%)	TI	Avg. vehicle number			Avg. total distance			Avg. insertion time		
			GVNS	Ref.	ARE (%)	GVNS	Ref.	ARE (%)	GVNS	Ref.	ARE (%)
R1	90	111.8	16.67	14.25	16.98	1409.06	1335.94	5.47	11.97	17.43	−31.33
	70	73.36	15.33	14.33	6.98	1379.52	1331.34	3.62	10.22	21.73	−52.97
	50	56.84	14.92	14.08	5.97	1333.75	1295.81	2.93	8.14	28.27	−71.21
	30	35.65	14.50	13.92	4.17	1292.46	1286.63	0.45	12.83	46.99	−266.25
	10	17.93	14.00	13.50	3.70	1252.82	1257.08	−0.34	14.20	67.99	−378.80
C1	90	0.00	10.67	10.78	−1.03	963.33	1039.77	−7.93	7.81	6.60	18.33
	70	0.00	11.33	10.78	5.10	1009.47	1031.68	−2.20	7.67	10.79	−28.92
	50	0.00	11.00	10.89	1.01	992.97	1001.18	−0.83	6.22	19.01	−67.28
	30	0.00	11.56	10.56	9.47	949.95	962.08	−1.28	9.13	28.02	−67.42
	10	0.00	10.56	10.56	0.00	898.3	895.77	0.28	13.74	45.40	−230.42
RC1	90	24.12	15.13	13.88	8.07	1559.47	1513.94	3.01	15.11	17.31	−12.71
	70	23.21	15.13	13.88	9.01	1560.91	1511.29	3.28	14.43	25.32	−43.01
	50	14.00	14.75	13.63	8.22	1526.21	1514.72	0.76	9.07	48.78	−81.41
	30	10.11	14.50	13.88	4.47	1489.33	1492.22	−0.19	13.17	45.26	−70.90
	10	9.25	14.00	13.38	4.63	1418.89	1436.23	−1.22	17.85	83.52	−367.90
Avg		25.08	13.60	12.83	5.78	1269.10	1260.38	0.39	11.44	34.16	−116.81

Table 6
Computational results of type 2 problems for various degrees of dynamism and infeasibilities.

Type	Dod (%)	TI	Avg. vehicle number			Avg. total distance			Avg. insertion time		
			GVNS	Ref.	ARE (%)	GVNS	Ref.	ARE (%)	GVNS	Ref.	ARE (%)
R2	90	0.00	4.00	3.55	12.68	1086.78	1047.82	3.72	16.47	13.20	24.77
	70	0.00	4.36	3.64	19.78	1078.03	1032.04	4.46	12.74	20.15	−36.77
	50	0.00	4.55	3.82	19.11	1071.83	1016.52	5.44	11.96	30.03	−60.17
	30	0.00	4.73	4.91	−3.67	1035.60	985.49	5.08	10.18	57.07	−82.16
	10	0.00	5.27	6.36	−17.14	1000.90	950.00	5.36	9.48	68.58	−86.18
C2	90	0.00	3.38	3.25	4.00	668.99	636.79	5.06	16.67	6.12	172.39
	70	0.00	3.38	3.13	7.99	672.95	636.47	5.73	14.03	10.01	40.16
	50	0.00	3.13	3.13	0.00	623.10	604.98	3.00	20.25	16.80	20.54
	30	0.00	3.25	3.63	−10.47	624.81	651.42	−4.08	34.82	29.87	16.57
	10	0.00	3.25	3.00	8.33	615.93	594.67	3.58	80.78	59.70	35.31
RC2	90	0.00	4.63	4.00	15.75	1275.93	1257.19	1.49	28.05	11.34	147.35
	70	0.00	5.13	3.88	32.22	1234.36	1239.46	−0.41	16.07	19.26	−16.56
	50	0.00	5.88	4.25	38.35	1200.26	1190.54	0.82	11.46	27.84	−58.84
	30	0.00	5.88	5.38	9.29	1172.33	1166.04	0.54	11.68	41.51	−71.86
	10	0.00	6.13	6.75	−9.19	1153.43	1103.30	4.54	13.27	55.55	−76.11
Avg		0.00	4.46	4.18	8.47	967.68	940.85	2.95	20.53	31.14	−2.10

Table 7
Computational results of type 1 problems for various degrees of dynamism and infeasibilities using averages.

Type	Dod (%)	TI	Avg. vehicle number			Avg. total distance			Avg. insertion time		
			GVNS	Ref.	ARE(%)	GVNS	Ref.	ARE(%)	GVNS	Ref.	ARE(%)
R1	90	123.11	16.65	14.25	16.84	1492.76	1335.94	11.74	13.72	17.43	−21.29
	70	86.69	16.32	14.33	13.86	1466.63	1331.34	10.16	11.54	21.73	−46.89
	50	60.69	15.94	14.08	13.24	1411.86	1295.81	8.96	9.82	28.27	−65.26
	30	40.18	14.99	13.92	7.72	1350.82	1286.63	4.99	13.75	46.99	−70.74
	10	10.50	14.66	13.5	8.60	1305.46	1257.08	3.85	14.56	67.99	−78.59
C1	90	0.00	11.37	10.78	5.48	1092.18	1039.77	5.04	9.73	6.60	47.42
	70	0.00	11.92	10.78	10.56	1150.27	1031.68	11.49	9.64	10.79	−10.66
	50	0.00	11.81	10.89	8.49	1129.58	1001.18	12.83	7.52	19.01	−60.44
	30	0.00	11.81	10.56	11.88	1081.52	962.08	12.41	11.29	28.02	−59.71
	10	0.00	11.36	10.56	7.53	986.99	895.77	10.18	15.77	45.40	−65.26
RC1	90	42.17	15.20	13.88	9.51	2776.89	1513.94	83.42	16.31	17.31	−5.78
	70	30.14	16.33	13.88	17.68	1687.54	1511.29	11.66	15.99	25.32	−36.85
	50	24.90	15.78	13.63	15.74	1629.64	1514.72	7.59	10.46	48.78	−78.56
	30	15.29	15.16	13.88	9.21	1571.12	1492.22	5.29	14.29	45.26	−68.43
	10	13.52	14.07	13.38	5.13	1459.85	1436.23	1.64	19.38	83.52	−76.80
Avg		29.81	14.22	12.82	10.77	1439.54	1260.38	13.42	12.92	34.16	−46.52

Table 8
Computational results of type 2 problems for various degrees of dynamism and infeasibilities using averages.

Type	Dod (%)	TI	Avg. vehicle number			Avg. total distance			Avg. insertion time		
			GVNS	Ref.	ARE (%)	GVNS	Ref.	ARE (%)	GVNS	Ref.	ARE (%)
R2	90	0	3.88	3.55	9.26	1181.31	1047.82	12.74	18.62	13.20	41.06
	70	0	4.22	3.64	16.05	1161.98	1032.04	12.59	13.87	20.15	−31.17
	50	0	4.49	3.82	17.56	1153.79	1016.52	13.50	12.81	30.03	−57.34
	30	0	4.77	4.91	−2.86	1112.92	985.49	12.93	12.34	57.07	−78.38
	10	0	5.49	6.36	−13.66	1054.82	950.00	11.03	10.99	68.58	−83.97
C2	90	0	3.66	3.25	12.56	749.33	636.79	17.67	17.44	6.12	184.97
	70	0	3.71	3.13	18.48	722.45	636.47	13.51	15.28	10.01	52.65
	50	0	3.53	3.13	12.62	670.23	604.98	10.79	20.32	16.80	20.95
	30	0	3.36	3.63	−7.48	670.88	651.42	2.99	35.76	29.87	19.72
	10	0	3.53	3.00	17.78	660.93	594.67	11.14	76.23	59.70	27.69
RC2	90	0	8.02	4.00	100.42	2032.46	1257.19	61.67	28.94	11.34	155.20
	70	0	4.94	3.88	27.36	1359.18	1239.46	9.66	17.27	19.26	−10.33
	50	0	5.39	4.25	26.86	1311.97	1190.54	10.20	12.36	27.84	−55.60
	30	0	5.83	5.38	8.43	1278.62	1166.04	9.65	12.71	41.51	−69.38
	10	0	6.01	6.75	−10.99	1240.55	1103.30	12.44	13.79	55.55	−75.18
Avg		0	4.72	4.18	15.49	1090.76	940.85	14.83	21.25	31.14	2.73

appear, and in this case we will try to minimize the total infeasibility in the plan of routes. Tables 5 and 6 give the comparison results in this case. Again, first column shows the set of test problem instances and the second column shows degrees of dynamism. The third column reports the average of the total infeasibility (in seconds) for each group of instances. The next columns show the average number of vehicles, the average total distances, and the average insertion times (in seconds) obtained using our method (*GVNS*) and the ones in Hong (2012) (*Ref.*). We also calculate the relative error (*ARE*(%)).

Note that this comparison in not really fair, since if we include all customers in the solutions, it will be reasonable that the number of vehicles and the total distance are higher than if we postpone some customers, which is what reference work Hong (2012) does.

As shown in the tables, the majority of times the reported total infeasibility is very small (a few seconds). Regarding total distance, we obtain results very close to the best ones (1% and 3% of average difference), but the number of times that we improve the results is less than in the previous experiments. However, as explained before, it has to be taken into account that it is logical that the routes containing all customers would involve longer distances. The same fact is observed with the number of routes. Nevertheless, our insertion times are again substantially lower than the best ones in the literature. It is noticeable that the total infeasibility for instances in Table 6 is always 0, because, as seen before, we did not postpone any customer service.

Once again, if instead of selecting the best results among the 15 executions to calculate the average values of each group of instances, we calculate the average of the 15 executions, we obtain Tables 7 and 8. In general, results have less quality than before, but notice that the works taken as references Hong (2012) and Xu et al. (2013) do not calculate these averages.

Table 9
Real instances with customer service postponing.

Dod (%)	NV	TD	First time	Total time	Post	Infeasibility
90	11.50	1180972.50	27.25	870.26	10.75	0.00
70	11.50	1073427.50	1360.99	1964.10	10.00	0.00
50	9.00	1066800.50	287.85	689.91	6.75	0.00
30	6.75	1021237.25	350.10	581.75	2.75	0.00
10	6.25	894760.25	670.85	745.18	2.00	0.00
Ave.	9.00	1047439.60	539.41	970.24	6.45	0.00

Table 10
Allowing infeasibility, but no customer service postponing.

Dod (%)	NV	TD	First time	Total time	Post	Infeasibility
90	12.50	1473052.50	34.79	924.97	0.00	18289.13
70	11.75	1446777.50	1710.79	1393.63	0.00	16013.00
50	9.75	1284227.50	271.30	730.83	0.00	10931.00
30	7.75	1094064.25	349.45	739.11	0.00	6533.88
10	6.25	958972.50	551.53	689.60	0.00	3388.38
Ave.	9.60	1251418.85	583.57	1095.63	0.00	11031.08

Table 11
Allowing infeasibility and customer service postponing.

Dod (%)	NV	TD	First time	Total time	Post	Infeasibility
90	11.25	1304612.50	31.98	1281.72	5.25	8802.63
70	10.50	1209942.50	1414.56	2021.65	4.50	6658.63
50	10.00	1149097.75	585.15	1047.78	4.50	4154.63
30	7.50	1091210.50	357.87	666.55	3.25	0.00
10	6.50	899648.75	580.63	677.87	1.25	812.75
Ave.	9.15	1130902.40	594.04	1139.11	3.75	4085.73

5.3. Computational results for real instances

Once we have verified that the algorithm is competitive using instances of the literature, it has been tested with the real instances of a company in the Canary Islands, $n1, n2, n3$, and $n4$. As for Solomon instances, 15 executions have been done for each instance and degree of dynamism, and the average of the best result for each instance has been calculated.

Three different experiments have been done using these real instances in order to see how customers priorities influence the results. In the first case, priorities allow to postpone any customer service if its insertion into routes involves some infeasibility. In the second case, customers priorities do not allow to postpone their service, so that infeasibilities are allowed if this is the only possibility when trying to insert them into a route. Finally, in the third case, priorities have been generated randomly, so that some customers can be postponed due to their low priorities, and other customers cannot. Thus, both postponed customers and infeasibilities appear.

Results in Tables 9–11 have been obtained, corresponding to the first, second and third experiment. The average over the four real instances is reported. First column reports the degree of dynamism. The next columns indicate the average number of vehicles, traveled distance, time for obtaining the initial static solution and insertion time. The last two columns show the average of postponed services and the average infeasibility (violations of customers time windows). As can be seen, the lowest number of vehicles and total traveled distance are obtained in the first experiment, because each customer which cannot be inserted without infeasibilities is postponed. An average of about 6 customers are postponed. However, in the second experiment it is exactly the opposite; the number of vehicles and the total distance increase because every customer is inserted, but the total infeasibility is really high. The results of the third experiment are in the middle situation, as some customers are postponed and other customers produce infeasibility when are inserted into routes. The number of postponed customers is about 3 and the total infeasibility is quite low. Finally, note that in these instances it was not required exceeding the working shifts of the vehicles.

In summary, we can conclude that the priorities of the customers significantly influence on the final plan of routes for the dynamic problem. Thereby, companies should carefully assign priorities to their customers in order to obtain the best balance between number of postponed services for the next day and the total infeasibility obtained.

6. Conclusions

A metaheuristic solution approach based on Variable Neighborhood Search (VNS) is proposed for solving a Dynamic Rich Vehicle Routing Problem with Time Windows (DRVRPTW). It combines a set of real-world constraints proposed by some companies in the Canary Islands, Spain. The designed algorithm manages two possibilities: rejecting the customers that cannot be feasibly inserted taking into account priorities, or permitting time windows infeasibilities in customers in order to provide a solution with all customers.

In order to assess the behavior of our approach, we have compared the obtained results with the best results in the literature using the standard test problem instances. In some cases, our

results are not only competitive with the related literature, but also even better. Moreover, our insertion times are substantially lower than the best ones.

Taking into account that the method proposed in this work has been developed to solve a real problem with a real set of constraints, it is not supposed to be the most competitive with the standard Solomon instances, which have other features. However, in that case, we have obtained results very close to the best ones in the literature.

Additionally, we propose solutions with infeasibilities in order to include all customers in the final solutions. In this case, logically, the total distance increases, but our results are still very close to the best ones in the literature. It is important to note that we are considering the time needed to insert any new dynamic customer in the plan, which can influence in the final results.

Finally, we have also analyzed the effect of the different restrictions in the final solutions using instances based on the real ones provided by a company. In this case, the importance of customers priorities on the final plan has become clear.

References

Bräysy, O. (2003). A reactive variable neighborhood search for the vehicle routing problem with time windows. *INFORMS Journal on Computing, 15*, 347–368.

Cassani, L., & Righini, G. (2004). Heuristic algorithms for the tsp with rear-loading. In *35th Annual Congress Italian Operations Research Society (AIRO XXXV)*. Lecce, Italy.

Chen, Z., & Xu, H. (2006). Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science, 40*(1), 74–88.

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

De Armas, J., Melián-Batista, B., Moreno-Pérez, J., & Brito, J. (2015). GVNS for a Real-World Rich Vehicle Routing Problem with Time Windows. *Engineering Applications of Artificial Intelligence* (accepted for publication).

Fleszar, K., Osman, I., & Hindi, K. (2008). A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research, 195*(3), 803–809.

Gendreau, M., & Potvin, J., (Eds.), (2004). *Transportation science* (Vol. 4).

Gendreau, M., Hertz, A., & Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research, 40*(6), 1086–1094.

Ghannadpour, S. F., Noori, S., Tavakkoli-Moghaddam, R., & Ghoseiri, K. (2014). A multi-objective dynamic vehicle routing problem with fuzzy time windows: Model, solution and application. *Applied Soft Computing, 14*, 504–527.

Ghiani, G., Guerriero, F., Laporte, G., & Musmanno, R. (2003). Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research*, 1–11.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research, 13*(5), 533–549.

Goel, A., & Gruhn, V. (2008). A general vehicle routing problem. *European Journal of Operational Research, 191*(3), 650–660.

Hansen, P., Mladenovic, N., & Moreno Perez, J. A. (2010). Variable neighbourhood search: Methods and applications. *Annals of Operations Research, 175*(1), 367–407.

Hemmelmayr, V., Doerner, K., & Hartl, R. (2009). A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research, 195*(3), 791–802.

Hoeller, H., Melian, B., & Voss, S. (2008). Applying the pilot method to improve VNS and GRASP metaheuristics for the design of SDH/WDM networks. *European Journal of Operational Research, 191*(3), 691–704.

Hong, L. (2012). An improved LNS algorithm for real-time vehicle routing problem with time windows. *Computers & Operations Research, 39*(2), 151–163.

Ibaraki, T., Imahori, S., Kubo, M., Masuda, T., Uno, T., & Yagiura, M. (2005). Effective local search algorithms for routing and scheduling problems with general time-window constraints. *Transportation Science, 39*(2), 206–232.

Ibaraki, T., Imahori, S., Nonobe, K., Sobue, K., Uno, T., & Yagiura, M. (2008). An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. *Discrete Applied Mathematics, 156*(11), 2050–2069.

Ichoua, S., Gendreau, M., & Potvin, J.-Y. (2007). Planned route optimization for real-time vehicle routing. In V. Zeimpekis, C. Tarantilis, G. Giaglis, & I. Minis (Eds.), *Dynamic fleet management. Operations research/computer science interfaces series* (Vol. 38, pp. 1–18). US: Springer.

Jozefowiez, N., Semet, F., & Talbi, E.-G. (2008). Multi-objective vehicle routing problems. *European Journal of Operational Research, 189*(2), 293–309.

Khouadjia, M., Sarasola, B., Alba, E., Jourdan, L., & Talbi, E. (2012). A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests. *Applied Soft Computing, 12*(4), 1426–1439.

Kytöjoki, J., Nuortio, T., Brysy, O., & Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research, 34*(9), 2743–2757.

Lackner, A. (2004). Selection meta-heuristics for dynamic vehicle routing problem (Dynamische Tourenplanung mit ausgewählten Metaheuristiken), no. 47, Göttinger Wirtschftinformatik.

Larsen, A. (2001). The dynamic vehicle routing problem, PhD thesis.

Larsen, A., Madsen, O., & Solomon, M. (2008). Recent developments in dynamic vehicle routing systems. In B. Golden, S. Raghavan, & E. Wasil (Eds.), *The vehicle routing problem: Latest advances and new challenges. Operations research/computer science interfaces* (Vol. 43, pp. 199–218). US: Springer.

Lund, K., Madsen, O., Rygaard, J. (1996). Vehicle routing problems with varying degrees of dynamism. Technical Report, IMM Institute of Mathematical Modelling.

Melian, B. (2006). Using memory to improve the VNS metaheuristic for the design of SDH/WDM networks. In: F. Almeida, M.J.B. Aguilera, C. Blum, J.M.M. Vega, M.P. Perez, A. Roli, M. Sampels (Eds.), *Hybrid metaheuristics, proceedings*, *Lecture notes in computer science* (Vol. 4030, pp. 82–93). (3rd International Workshop on Hybrid Metaheuristics, Gran Canaria, SPAIN, OCT 13-14, 2006).

Melian-Batista, B., De Santiago, A., AngelBello, F., & Alvarez, A. (2014). A bi-objective vehicle routing problem with time windows: A real case in Tenerife. *Applied Soft Computing* (17), 140–152.

Montemanni, R., Gambardella, L., Rizzoli, A., & Donati, A. (2005). Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization, 10*(4), 327–343.

Moreno-Vega, J. M., & Melian, B. (2008). Introduction to the special issue on variable neighborhood search. *Journal of Heuristics, 14*(5), 403–404.

Or, I. (1976). *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. Northwestern University.

Pillac, V., Gendreau, M., Gueret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research, 225*(1), 1–11.

Polacek, M., Hartl, K., Doerner, K., & Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of Heuristics, 10*(6), 613–627.

Psaraftis, H. (1980). A Dynamic-programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science, 14*(2), 130–154.

Repoussis, P., Paraskevopoulos, D. C., Tarantilis, C., & Ioannou, G. (2006). A reactive greedy randomized variable neighborhood tabu search for the vehicle routing problem with time windows, 4030, pp. 124–138.

Rizzoli, A., Montemanni, R., Lucibello, E., & Gambardella, L. (2007). Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence, 1*, 135–151.

Schmid, V., Doerner, K. F., & Laporte, G. (2013). Rich routing problems arising in supply chain management. *European Journal of Operational Research, 224*(3), 435–448.

Solomon, M. (1987). Algorithms for the vehicle-routing and scheduling problems with time window constraints. *Operations Research, 35*(2), 254–265.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J.-Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science, 31*(2), 170–186.

Vidal, T., Crainic, T., Gendreau, M., & Prins, C. (2013). Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*.

Xu, Y., Wang, L., & Yang, Y. (2013). Dynamic vehicle routing using an improved variable neighborhood search algorithm. *Journal of Applied Mathematics, 2013*, 12.