J Heuristics (2013) 19:103–127
DOI 10.1007/s10732-011-9159-1

Designing routes for WEEE collection: the vehicle routing problem with split loads and date windows

Julio Mar-Ortiz · José Luis González-Velarde · Belarmino Adenso-Díaz

Received: 10 March 2010 / Accepted: 7 February 2011 / Published online: 26 February 2011

Abstract This paper presents an integer programming model and describes a GRASP based algorithm to solve a vehicle routing and scheduling problem for the collection of *Waste of Electric and Electronic Equipment* (WEEE). The difficulty of this problem arises from the fact that it is characterized by four variants of the vehicle routing problem that have been studied independently in the literature, but not together. The experimental analysis on a large set of randomly-generated instances shows the good performance of the proposed algorithm. Moreover, computational results using real data show that the method outperforms real existing approaches to reverse logistics.

Keywords Vehicle routing problem · Metaheuristics · Reverse logistics · Date-windows · Split-loads

Electronic supplementary material The online version of this article (doi:10.1007/s10732-011-9159-1) contains supplementary material, which is available to authorized users.

J. Mar-Ortiz
Faculty of Engineering, Universidad Autónoma de Tamaulipas, 89140 Tampico, Mexico
e-mail: jmar@uat.edu.mx

J. Mar-Ortiz · J.L. González-Velarde (✉)
Tecnológico de Monterrey, 64849 Monterrey, Nuevo León, Mexico
e-mail: gonzalez.velarde@itesm.mx

B. Adenso-Díaz
Engineering School, Universidad de Oviedo, 33203 Gijón, Spain
e-mail: adenso@epsig.uniovi.es

1 Introduction

The vehicle routing problem (VRP) is one of the most studied problems in the combinatorial optimization field and in the logistics literature. There are different variants of the VRP that can be characterized by the type of fleet, the number of depots, or the type of operations involved, among others. While, in recent years, there has been an increasing interest towards so-called rich VRP models that include important issues arising from real-world applications (Battarra et al. 2009), the literature on vehicle routing for reverse logistics is still very scarce. Most of the papers address specific problems and case studies (Schultmann et al. 2006; Blanc et al. 2006; Kim et al. 2009), while theoretical contributions mainly deal with the VRP with simultaneous pick-up and delivery, or the VRP with backhauls.

This contribution focuses on the study of a VRP for reverse logistics. It specifically addresses the problem of designing routes for the collection of Waste of Electric and Electronic Equipment (WEEE). The motivation for this study is twofold. First, the fact that the majority of European countries have recently implemented the European Union Directive 2002/96/EC on WEEE (European Parliament the Council and the Commission 2003), whose general purpose is to prevent the creation of electrical and electronic waste and to promote reuse, recycling and the other forms of recovery in order to reduce final disposal. Second, a real-world case study performed by the authors in the northwest of Spain (Mar-Ortiz et al. 2010), where a national WEEE Collective Management System comprising of 5 depots has to collect the WEEE from a set of 707 geographically-dispersed stores. Each store has a given capacity (directly related to sales volume) to collect WEEE. In turn, every store has in its warehouse a certain space allotted for the WEEE collected. Whenever this space is about to be saturated, the store manager calls the coordination call center for these items to be taken off by the corresponding depot. Each depot holds a heterogeneous fleet of capacitated vehicles, which are used in the collection of WEEE. The collection should be completed within a period of one week after receiving the call from the store.

The aim of this research is to design a Greedy Randomized Adaptive Searching Procedure (GRASP) algorithm to solve a real-world problem where a fixed and heterogeneous fleet of capacitated vehicles with special features is used in the collection of WEEE from a set of customers. Each vehicle can at most perform a predefined number of trips without the given maximum operation time being exceeded. The amount of end-of-life items to be collected (hereafter referred to as the demand) of each customer can be greater than the capacity of the largest vehicle in the fleet. As a result the demand of a single customer may be split to be carried out by more than one route. Thus, each customer can be visited more than once on the same day. Collection orders to customers are triggered by a call from them. With each order there is an associated demand to be met and a range of dates within the collection should take place. This collection scheme is similar to the one used in the Netherlands for the collection of white goods (Beullens et al. 2004), and is characteristic of several reverse logistics systems. Thus, the proposed algorithm may be easily adapted to solve other related collection problems beyond this specific case study.

The problem depicted above is characterized by four variants of the VRP that have been independently studied in the logistics literature, including: (1) the use of

a heterogeneous fleet of vehicles, (2) customers with a high demand that may be split to fit the vehicle capacity, (3) the fact that the same vehicle may be assigned to more than one route, and (4) the agreement of a time interval for visiting customers. These characteristics make the problem complex and interesting, while complicating its classification within the standard VRP literature (Eksioglu et al. 2009). We label this variant of the VRP as the *Vehicle Routing Problem with Split Loads and Date Windows* (VRPSLDW).

The remainder of this section analyzes and discusses the related literature. In Sect. 2 the problem is described, with special emphasis on its mathematical formulation as an integer programming model. Section 3 presents the solution algorithm proposed, while Sect. 4 presents the experimental study of the algorithm. After tuning the parameters and analyzing the performance of the algorithm in different problem instances, the results of the case study are discussed in Sect. 5. Finally, Sect. 6 provides the conclusions of the study.

1.1 Related literature

The variants that characterize the problem addressed in this research have been previously cited. Feature (1) mentioned above is referred to as the heterogeneous fleet vehicle routing problem (HF-VRP), of which there are three variants, two of them related to the number of vehicles of each type: limited or unlimited; and the third one refers to the accessibility restrictions of vehicles for customers. Among the first authors undertaking the HF-VRP we refer to Golden et al. (1984). Recently Imran and Wassan (2009) proposed an adaptation of the Variable Neighborhood Search metaheuristic embedded with two variants of the Dijkstra algorithm for the general problem. The third variant of the HF-VRP is referred to as the Site Dependent VRP (SD-VRP), and was introduced by Nag et al. (1988). Feature (2) is referred to as vehicle routing problem with split loads (VRP-SL), which was introduced by Dror and Trudeau (1989) detailing its heuristic properties. They showed that the splitting of loads may result in savings, for both the total travelled distances as well as in the number of vehicles used. Mitra (2008) indicates that by allowing the loads to be split, still better results may be obtained than those achieved by the combined objective of minimizing the fixed costs of load and costs associated with the routes of vehicles. Recently, Aleman et al. (2010) has proposed a constructive algorithm for the VRP-SL based on a concept called the route angle control measure, which was integrated into an iterative approach using adaptive memory concepts, and embedded into a variable neighborhood descent process, obtaining favorable results compared with existing approaches.

Feature (3) refers to the contributions in the vehicle routing with a multiple use of the same vehicles. Despite the standard VRP definition implicitly assumes that each vehicle is used only once beyond the planning horizon, in reality once the routes have been designed, it is possible to assign several of them to the same vehicle in such a way that the vehicles can operate feasibly within the planning horizon. The problem of designing routes with multiple trips is known in the literature as the multi-trip VRP (MT-VRP) and was introduced by Taillard et al. (1996) and Brândao and Mercer (1998).

The problem featured by (4) does not establish periodic visits as the periodic VRP. Nevertheless, every time the store calls the collective management system, the interval of days within which the collection should take place is previously agreed. Thus, this feature can be understood as a date window that provides route flexibility. The dissimilarity with the VRP with Time Windows (VRPTW) should be noted. This is a generalization of the classical VRP wherein each customer i must be served within its corresponding time window $[e_i, l_i]$. Hence, the routes should be designed in such a way for each customer to be visited only once by exactly one vehicle within the given interval of time. On the other hand, the VRP with Date Windows (VRPDW) is less restrictive than the VRPTW with respect to the time of visits. However the VRPDW is conducted at a broader level in a period T, without being a periodic VRP (each customer must be visited any time in a predefined interval of days). For example, in the VRPTW two different customers $\{i, j\}$ cannot be served at the same time by the same vehicle. Nevertheless, in the VRPDW two different customers can be served by the same vehicle on the same day. Nor is the VRPDW periodic, given that what is collected on day t is dependent on what was collected on day $t - 1$.

The literature review shows that features (1) and (2) were studied in a joint form by Belfiore and Yoshida-Yoshizaki (2008) who proposed a Scatter Search algorithm to solve a real-world problem with a heterogeneous fleet of vehicles with time windows and split loads. The research that, on its own, resembles the assumption described by the feature (4) more is the one developed by Alvarez et al. (2009). They consider a VRP in which, despite the existence of a delivery due date, there is certain flexibility to perform the delivery some days before. Features (1), (3) and (4) were addressed by Alonso et al. (2007) who introduced the Site-Dependent Multi-Trip Periodic VRP (SDMTVRP). Nevertheless, to the best of our knowledge these four features have not been studied all together.

2 Problem definition and formulation

Formally speaking the mathematical formulation of the problem requires the definition of a directed graph $G = (V, E)$ where $V = \{v_0, v_1, \ldots, v_m\}$ is a set of vertices, and $E = \{(v_i, v_j) : i \neq j\}$ is a set of arcs connecting the vertices. Vertex v_0 denotes the depot where a heterogeneous fleet of $K = \{1, \ldots, k\}$ vehicles with capacity Q_k are stationed, while the remaining m vertices of V represent the customers to be visited. For every pair of vertices $i, j \in \{V : i \neq j\}$ the distance d_{ij} between them is known. For each vehicle k we know the maximum load capacity, the set of customers that vehicle k may visit (because of street access restrictions), and the variable travelling cost of vehicle k. For the collection, the independent customers make use of a call system with a timely collection guarantee. Each collection order predefines the quantity of items w_i to be collected from customer i and the range of dates within which the collection should take place. The time interval to carry out the collection from a single customer is specified by its date window $[s_i, f_i]$, where s_i and f_i are respectively the first and last days within which the collection must take place, such that $s_i \leq f_i$. For all customers the service time μ_i is assumed to be constant, since each customer is usually served in a full truck load, being visited when it has at hand

Table 1 Notation used in the mathematical model, where card(X) represents the cardinality of the set X

Sets:	
V	Set of vertices, representing the union of all customers allotted to the depot plus the depot itself, where $\text{card}(V) = m + 1$.
E	Set of arcs (v_i, v_j), $\text{card}(E) = n$.
T	Set of days in the planning horizon, $t = 1, \ldots, T$.
K	Set of available vehicles, $k = 1, \ldots, K$.
Parameters:	
w_i	Amount of items to be collected (demand) at customer $i \in V \backslash \{0\}$.
ε	Minimum load to be collected at any customer i, to justify its visit.
d_{ij}	Traveling distance (km) from vertex i to vertex j.
t_{ij}	Traveling time (hrs) from vertex i to vertex j.
μ_i	Service time (hrs) at vertex $i \in V$.
Q_k	Capacity of vehicle k.
L_k	Maximum operation time for vehicle k per day.
λ_k	Traveling cost per kilometer for vehicle k.
R	Maximum number of trips that any vehicle may perform per day.
A	Binary matrix, where $a_{it} = 1$ means that customer i must be visited within the set of days defined by its date windows $\{t \in T : s_i \leq t \leq f_i\}$.
B	Binary matrix, where $b_{ik} = 1$ means that vehicle k is feasible for customer i, and 0 otherwise. It should be noted that $b_{0k} = 1$ for all $k \in K$.
Binary decision variables $O(KRTn)$:	
$y_{ij}^{(tkr)}$	1 if arc $(i, j) \in E$ is transverse within the optimal solution by the trip r of vehicle k on day t, and 0 otherwise. In this sense, a route is defined by a set of three attributes (tkr).
Integer decision variables $O(m(KRT + 1))$:	
$x_i^{(tkr)}$	Amount of items collected at customer i on route (tkr).
σ_i	Amount of items not collected from customer i, due to its economical infeasibility.
Continuous decision variables $O(KRTm)$:	
$u_i^{(tkr)}$	Cumulative load in route (tkr) after visiting customer i.

a sufficient number of items to be collected. For the sake of real practice, when the number of items to be collected from any customer i is less than a constant factor ε, those items are not collected and are labeled as backorders to be considered in a following request. The notation used in the mathematical formulation is described in Table 1.

The optimization problem is to determine which customers are served by each vehicle, what route the vehicle will follow to serve those customers assigned and how much demand each vehicle will collect from each customer, while minimizing the routing costs. The mathematical model for the VRPSLDW is given by:

$$\text{Minimize} \quad \sum_{i \in V} \sum_{j \in V} \sum_{t \in T} \sum_{k \in K} \sum_{r=1}^{R} \lambda_k \cdot d_{ij} \cdot y_{ij}^{(tkr)} \tag{1}$$

subject to:

$$\sum_{i \in V \setminus \{j\}} \sum_{r=1}^{R} y_{ij}^{(tkr)} \leq a_{jt} \cdot b_{jk} \quad \forall j \in V \setminus \{0\},\ t \in T,\ k \in K \tag{2}$$

$$\sum_{j \in V \setminus \{0\}} y_{0j}^{(tkr)} \leq 1 \quad \forall t \in T,\ k \in K,\ r = 1, \ldots, R \tag{3}$$

$$\sum_{i \in V \setminus \{j\}} y_{ij}^{(tkr)} = \sum_{i \in V \setminus \{j\}} y_{ji}^{(tkr)} \quad \forall j \in V,\ t \in T,\ k \in K, r = 1, \ldots, R \tag{4}$$

$$x_j^{(tkr)} \leq w_j \cdot \sum_{i \in V \setminus \{j\}} y_{ij}^{(tkr)} \quad \forall j \in V \setminus \{0\},\ t \in T,\ k \in K,\ r = 1, \ldots, R \tag{5}$$

$$x_j^{(tkr)} \geq \varepsilon \cdot \sum_{i \in V \setminus \{j\}} y_{ij}^{(tkr)} \quad \forall j \in V \setminus \{0\},\ t \in T,\ k \in K,\ r = 1, \ldots, R \tag{6}$$

$$y_{ij}^{(t,k,r)} \geq y_{ij}^{(t,k,r+1)} \quad \forall i, j \in V \setminus \{0\},\ t \in T,\ k \in K, r = 1, \ldots, R-1 \tag{7}$$

$$\sum_{t \in T} \sum_{k \in K} \sum_{r=1}^{R} x_i^{(tkr)} + \sigma_i = w_i \quad \forall i \in V \setminus \{0\} \tag{8}$$

$$\sum_{i \in V} \sum_{j \in V} \sum_{r=1}^{R} (t_{ij} + \mu_j) \cdot y_{ij}^{(tkr)} \leq L_k \quad \forall t \in T,\ k \in K \tag{9}$$

$$u_j^{(tkr)} \geq u_i^{(tkr)} + x_j^{(tkr)} + Q_k \cdot (y_{ij}^{(tkr)} - 1)$$
$$\forall i, j \in V \setminus \{0\} :\ i \neq j,\ t \in T, k \in K,\ r = 1, \ldots, R \tag{10}$$

$$x_i^{(tkr)} \leq u_i^{(tkr)} \leq Q_k \quad \forall i \in V \setminus \{0\},\ t \in T,\ k \in K,\ r = 1, \ldots, R \tag{11}$$

$$x_i^{(tkr)} \geq 0, u_i^{(tkr)} \geq 0,\ 0 \leq \sigma_i \leq \varepsilon - 1$$
$$\forall i \in V,\ t \in T,\ k \in K,\ r = 1, \ldots, R \tag{12}$$

$$y_{ij}^{(tkr)} \in \{0, 1\} \quad \forall i, j \in V,\ t \in T,\ k \in K,\ r = 1, \ldots, R \tag{13}$$

The objective function (Eq. 1) minimizes the total routing cost in all journeys made by the fleet of vehicles. Equations 2, 3 and 4 provide the connectivity and continuity constraints. Equation 2 states that a customer j can be reached directly just from a single customer $i \neq j$, on a trip r of a feasible route (t, k, r). In this sense, a *route* is defined by a set of three attributes: the day t, the vehicle k and the trip r. Equation 3 guarantees that each route (t, k, r) may be performed just once at most. Equation 4 establishes the continuity conditions and ensures that the vehicle k arriving at a given customer j in its trip r in day t, is the same vehicle leaving such a customer. Equations 5 and 6 specify the collection bounds for each customer on any route. Equation 5 states that the amount of items collected from a customer j by route (t, k, r) does not exceed the customers' demand, whilst Eq. 6 provides that at least a constant factor of ε items must be collected by a given route leaving the depot.

Equation 7 guarantees the proper use of trips in every route. It establishes that every day the trip r of a vehicle k must be performed before its trip $r+1$. Moreover, constraints (Eqs. 6 and 7) strengthen the formulation by restricting duplicate solutions, which reduce the number of nodes in the branch-and-bound tree. Equation 8 ensures the collection of a customer's demand in one or more routes, or its consideration as a backorder. Equation 9 sets the maximum service time for each vehicle, whereas Eqs. 10 and 11 provide both the capacity restrictions and the sub-tour elimination constraints. Finally, Eqs. 12 and 13 deal with the nature of the decision variables. Note that although $x_i^{(tkr)}$ and σ_i are declared as continuous, the formulation forces them to take integer values.

Problem complexity The problem modeled above can be characterized as a new variant of the VRP that incorporates several elements that have been independently studied in the literature. This problem is NP-Hard since it is a variant of the VRP that incorporates a larger number of decision elements. To show the difficulty of the problem, a trial example with just 10 customers totalizing 607 units of demand was solved: a set of 3 vehicles with capacities of 25, 30 and 50 units respectively, was allotted to the depot. Each vehicle can carry at most 3 trips per day, without exceeding 7 working hours per day in a period of week. This sample problem results in a mixed integer linear programming model with 7,570 variables (6,300 of which are binary) and 21,333 constraints. The problem was modeled on AMPL and solved by CPLEX v.11.1 using an Acer PC with an AMD Athlon X2 Processor running at 1.90 GHz and 2 GB RAM. After 3 hours of computing time, the best solution found was 1,229.66, with a relative best node deviation of 0.3008, and an absolute relative best node deviation of 369.981.

3 Heuristic solution procedure

In this section we introduce a GRASP based heuristic to solve the Vehicle Routing Problem with Split Loads and Date Windows (VRPSLDW) described above. GRASP was introduced in Feo and Resende (1989) and later formalized in Feo and Resende (1995). The motivation for using GRASP in this particular application is based on the fact that there is a natural form to define a priority function to select the customer to be inserted in the initial solution: by linking the customer demand and the available time for its collection.

3.1 General outline of the metaheuristic procedure

When designing an algorithm it is important to define a good structure for encoding the solution. Computationally speaking, in our particular case the simplest structure to consider for representing the solution is a LOAD matrix (see example in Fig. 1) which has as many rows as the total number of routes ($K \times R \times T$) and as many columns as the number of customers to serve (m). Every route π is defined by the set of three attributes: the day t, the vehicle k and the trip $r(\pi_{kr}^t)$. Each cell (π, i) specifies the amount of demand $l_{\pi i}$ collected on route π from customer i. Since every row in

		Customers											
Route		1	2	3	4	5	6	7	8	9	10	$\bar{Q}_k$	Q_k
A	π^1_{11}				9		16					0	25
B	π^1_{31}						40					10	50
C	π^1_{32}			50								0	50
D	π^1_{33}					50						0	50
E	π^2_{11}	8	14									3	25
F	π^2_{12}			8	17							0	25
G	π^2_{31}										50	0	50
H	π^2_{32}			50								0	50
I	π^3_{11}			25								0	25
J	π^3_{21}							29				1	30
K	π^3_{31}					50						0	50
L	π^3_{32}						50					0	50
M	π^4_{11}								23			2	25
N	π^4_{12}								25			0	25
O	π^5_{11}								25			0	25
P	π^5_{31}					46						4	50
Q	π^6_{11}									10	11	4	25
	σ_i	0	0	1	0	0	0	0	0	0	0		
	ω_i	8	14	134	26	146	106	29	73	10	61		

Fig. 1 Solution to the trial example. *Column* $\bar{Q}_k$ states for the remaining capacity of the vehicles used in their corresponding routes and σ_i the amount of items not collected from customer i. *Gray cells* indicate that the route π is penalized for a given customer i; i.e., those days are out of the specific date window for the customer. Only routes with loads assigned are shown

LOAD corresponds to a route and every column corresponds to a customer, a feasible solution must satisfy that: (a) for each row its sum must not exceed the capacity of the vehicle k used in route π, (b) the cumulative sum for the traveling time in a subset of routes using a vehicle k in a given day t must not exceed L_k, and (c) for each column its sum must be at most equal to the demand of each customer w_i.

Although the mathematical model requires the collection to be done between s_i and f_i, the algorithm solution relaxes this feature, allowing the collection after f_i, but incurring in a penalty cost for each day after the collection date promised. Therefore, the use of an appropriate data structure is required to identify when a route is feasible, unrestricted and non-penalized for each customer i. A route π is *feasible* for customer i if and only if the vehicle allotted to the route may access to the customer's location. On its own, a route π is *unrestricted* for customer i if both the remaining capacity of the route $\bar{Q}_\pi$ and the maximum operation time L_k of all routes using the same vehicle k on the same day t are not exceeded when the customer i is inserted in route π. Finally, a route π is *non-penalized* for customer i if the day on which the

		Customers									
Route		1	2	3	4	5	6	7	8	9	10
A	π^1_{11}				1		2				
B	π^1_{31}						1				
C	π^1_{32}			1							
D	π^1_{33}					1					
E	π^2_{11}	2	1								
F	π^2_{12}			2	1						
G	π^2_{31}										1
H	π^2_{32}			1							
I	π^3_{11}			1							
J	π^3_{21}							1			
K	π^3_{31}					1					
L	π^3_{32}						1				
M	π^4_{11}								1		
N	π^4_{12}								1		
O	π^5_{11}								1		
P	π^5_{31}					1					
Q	π^6_{11}									1	2

Fig. 2 ROUTE matrix for the trial example. *Each cell* (π, i) indicates the order in which the customer i is visited in a route π. This figure only shows those routes (*rows*) used in a particular solution

route has been allotted is within the date window $[s_i, f_i]$ for customer i. A feasible solution s has a set of feasible routes $\Pi = \{\pi_1, \ldots, \pi_\upsilon\}$ associated.

The simplest structure considered to represent the set of routes corresponds to another matrix ROUTE (see Fig. 2) with the same dimensions as the LOAD matrix. Each row in the ROUTE matrix represents a route π, each column a customer i and each cell (π, i)(with $l_{\pi i} \geq 1$) indicates the order in which the customer i is visited on route π. Given a new assignment in the LOAD matrix, the cost of the modified route is evaluated by solving the corresponding TSP (*Traveling Salesman Problem*) in ROUTE. Each route $\pi \in \Pi$ has an associated travel time h_π related to both the number of customers visited in the route and the order in which they are visited. Since the traveling time sum across the set of routes that uses the same vehicle k on the same day t should not exceed L_k (i.e. $\sum_{\pi \in \Pi | \pi = \pi_k} h_\pi \leq L_k$) both matrix LOAD and ROUTE are interdependent.

3.2 Greedy randomized construction

The construction mechanism at each iteration aims to allocate the largest amount of unsatisfied demand from customer $i \in V \backslash \{0\}$ to a single route $\pi \in \Pi$ such that π is feasible, unrestricted and non-penalized for customer i.

The basic principle for the construction phase aims to fulfill the routes as much as possible and split the demand as little as possible. The constructive procedure starts with an empty solution, and the customers demand is iteratively distributed among the routes according to the greedy randomized construction scheme. When the total demand of each customer has been distributed among the routes or the pending demand to assign of each customer is smaller than a value ε, the constructive phase finishes and the local search procedure is initiated. To select the next customer i to be included in the solution, we make use of a greedy function $\phi(i)$ which relates the pending demand $\overline{w}_i$ to satisfy from each customer i and the time available to perform the service. The value $\phi(i)$ measures the critical ratio for customer i, so the higher it is the most urgent its service is. This ratio is computed as:

$$\phi(i) = \frac{\overline{w}_i}{slack_i} \tag{14}$$

where $slack_i = f_i - s_i$ represents the available time to carry out the collection from customer i. If $f_i = s_i$, then $\phi(i) = INF$.

The detailed procedure is depicted in Fig. 3. If there is at least one customer i with $\phi(i) = INF$, then the Restricted Candidate List (RCL) is exclusively formed by such customers. Otherwise, a customer i with $\phi(i) > 0$ belongs to the RCL if: $\phi(i) \geq \Phi_{\max} - \alpha \cdot (\Phi_{\max} - \Phi_{\min})$, where $\Phi_{\min}$ and $\Phi_{\max}$ are respectively the lowest and highest value obtained by the greedy function, and α $(0 \leq \alpha \leq 1)$ is a parameter that controls the degree of randomness allowed. The selection of a customer i within the RCL is done randomly. Subsequently, a call to the AssignLoad procedure is performed. Its main function consists of finding a feasible route π for the selected customer i, to whom the largest amount of pending load $(\overline{w}_i)$ for such customer can be assigned. With this aim, we define $\Pi^i \subseteq \Pi$ as the subset of feasible, unrestricted and non-penalized routes for customer i, and the following scenarios are considered: (a) if $|\Pi^i| = 0$, then the date window for customer i is artificially augmented by one day, without affecting $slack_i$; (b) if $|\Pi^i| = 1$, then $x_{\pi i} = \min\{\bar{Q}_\pi, \overline{w}_i\}$, where $x_{\pi i}$ represents the load of customer i to be assigned to route π; and (c) if $|\Pi^i| > 1$, then customer i is assigned to route π capable of carrying the largest load (in the case of ties, customer i is assigned to the route π where $H_\pi = \bar{Q}_\pi - x_{\pi i}$ is minimized). Finally, customer i is inserted into the route π, and both matrix ROUTE and LOAD are updated. The remaining load space $\bar{Q}_\pi$ in route π and the pending load to satisfy $\overline{w}_i$ for customer i are also updated. When updated, if the pending load to satisfy for customer i is less than or equal to ε, then customer i is removed from the set of customers with pending load U. As pointed out above, this is a common practice in any routing system. Where a trip to collect a very small number of devices is never justified, however, this is considered in a following request.

3.3 Local search

The local search procedure is carried out by manipulating the allocation of the portion of demand $l_{\pi i}$ in the LOAD matrix and updating the routes involved in ROUTE. This procedure is made up of two phases. Phase I verifies if penalties exist by serving the customers on a penalized route, and seeks to eliminate them. Phase II explores

```
  procedure GreedyRandomizedConstruction( )
  Input: α, LOAD, ROUTE.
  Output: Set of feasible routes.
0  ϖ_i ← ω_i  ∀i ∈ V\{0};  U ← V;
1  while U ≠ ∅ do
2    Compute φ(i) as in equation (14) for all i ∈ U;
3    Φ_min ← min_i{φ(i)|i ∈ U};  Φ_max ← max_i{φ(i)|i ∈ U};
4    if Φ_max < INF then RCL ← {i ∈ U|φ(i) ≥ Φ_max − α · (Φ_max − Φ_min)};
5    else RCL ← {i ∈ U|φ(i) = INF};
6    Randomly select i from the RCL;
7    route ← null;
8    x_πi ← AssignLoad(i, route, DP, ϖ_i);
9    if x_πi > 0 then
10     if l_πi = 0 then
11       Update ROUTE matrix;
12       Update routing costs and traveling times;
13     end if;
14     Update LOAD matrix: l_πi ← l_πi + x_πi;
15     Q̄_π ← −x_πi;
16     ϖ_i ← ϖ_i − x_πi;
17     if ϖ_i ≤ ε then U ← U\{i}; end if;
18   end if;
19 end while;
20 Return LOAD, ROUTE, routes_cost, penal_cost.
  end procedure GreedyRandomizedConstruction
```

Fig. 3 Greedy randomized construction procedure phase for the VRSPSLDW

four neighborhoods of the solution space in order to reduce the routing costs. All the moves described in the VRP-SL literature analyzed so far are included in our exploration. The moves are described below.

Phase I of local search (elimination of penalties) The moves considered in this phase of the local search are: m_{I1} the insertion move, and m_{E1} the exchange move. The neighborhoods created by these moves are completely explored under the best improving strategy. Given an initial solution, a greedy local search is implemented by each time choosing the best possible move between both neighborhoods. To detail both moves, the definition of the following sets is required: let P be the set of penalized customers i and C the set of customers $j \neq i$ that have an assignation in some of the routes Π for which i is non-penalized and the insertion of i would be feasible and unrestricted.

1. *Insertion Move 1* (m_{I1}). This move comprises two steps. In the first step, for each customer $j \in C$ with $l_{\pi j} > 0$, the portion $x_{\pi j}^{\hat{\pi}}$ of load allotted to π that can be assigned to any other feasible, non-penalized and unrestricted route $\hat{\pi}$ for j is computed, such that $x_{\pi j}^{\hat{\pi}} = \min\{\bar{Q}_{\hat{\pi}}, l_{\pi j}\}$. Let $X_j^{\hat{\pi}} = \max_{\hat{\pi}}\{x_{\pi j}^{\hat{\pi}}\}$ the maximum load for customer j that can be moved from π to $\hat{\pi}'$, and $X^* = \max\{X_j^{\hat{\pi}}\}$. The following options are considered: (a) if $X^* = 0$, then it is not possible to reduce the load in penalization by means of the insertion move, and it is passed on to the next move, (b) if $X^* > 0$, then move the $x_{\pi j}^{\hat{\pi}}$ units from the route π of customer j that gave rise to X^*, to the route $\hat{\pi}$. In the second step we define $E_\pi = \bar{Q}_\pi +$

```
procedure LocalSearchPhaseII( )
  Input: LOAD, ROUTE.
  Output: Improved cost solution.
0   do {
1     local_optima ← true;
2     x^insert ← Best move obtained from insertion move 2;
3     x^interchange ← Best move obtained from interchange move;
4     x^combine ← Best move obtained from combine move;
5     x^elimination ← Best move obtained from elimination move;
6     m* ← Best move among all neighborhoods;
7     if ∃ m* then
8       Update LOAD and ROUTE in according to m*.
9       Update routes_cost;
10      Local_optima ← false;
11    end if;
12  } while local_optima == false
13  Return improved LOAD, ROUTE.
  end procedure LocalSearchPhaseII
```

Fig. 4 Phase II of the local search for the VRSPSLDW

X^* as the space that would remain free in the route π after removing X^* units. Then for each customer $i \in P$ such that the route π is feasible, we compute $x_{ri}^{\pi} = \min\{l_{ri}, E_{\pi}\}$, the maximum amount of penalized load for customer i that can be moved from its current penalized route r to the non-penalized route π, and its corresponding pending load be still penalized $CP_i = l_{ri} - x_{ri}^{\pi}$. The best move in this neighborhood is the one in which x_{ri}^{π} is maximum and CP_i is minimum.

2. Exchange Move (m_{E1}). For each customer $i \in P$ visited on a route π, we consider the exchange of $x_{\pi i}^{\hat{\pi} j} = \min\{l_{\pi i}, l_{\hat{\pi} j}\}$ units with all the routes $\hat{\pi} \in \Pi$ of each customer $j \in C$ such that π is a feasible and non-penalized route for j. With every move the corresponding CP_i is computed: $CP_i = l_{\pi i} - x_{\pi i}^{\hat{\pi} j}$. The best move in this neighborhood is the one in which $x_{\pi i}^{\hat{\pi} j}$ is maximum and CP_i is minimum.

Phase II of local search (cost minimization) The moves considered in this second phase of the local search are: m_{I2} the insertion move, m_{E2} the interchange move, m_{C} the combined route move, and m_{E} the route elimination move. The neighborhoods created by these moves are also completely explored via the best improving strategy. Given an initial solution, a greedy local search is implemented by choosing each time the best possible move among all neighborhoods (see pseudo-code in Fig. 4). Thus, the order in which the neighborhoods are explored does not have any influence on the algorithmic performance.

1. Insertion Move 2 (m_{I2}). This move is based on removing the largest portion of the load that a customer i has assigned to a route π, and inserting it into another feasible, non-penalized, and unrestricted route $\hat{\pi}$ for customer i. The portion of load to be moved is given by $x_{\pi i}^{\hat{\pi}} = \min\{\bar{Q}_{\hat{\pi}}, l_{\pi i}\}$. If $x_{\pi i}^{\hat{\pi}} = l_{\pi i}$ implies that all the load of customer i on route π was removed and reassigned into another single route $\hat{\pi}$. See example in Fig. 5.
2. *Interchange Move* (m_{E2}). This move is based on removing a customer from one route and inserting it into another route. For each pair of different customers $i \neq j$,

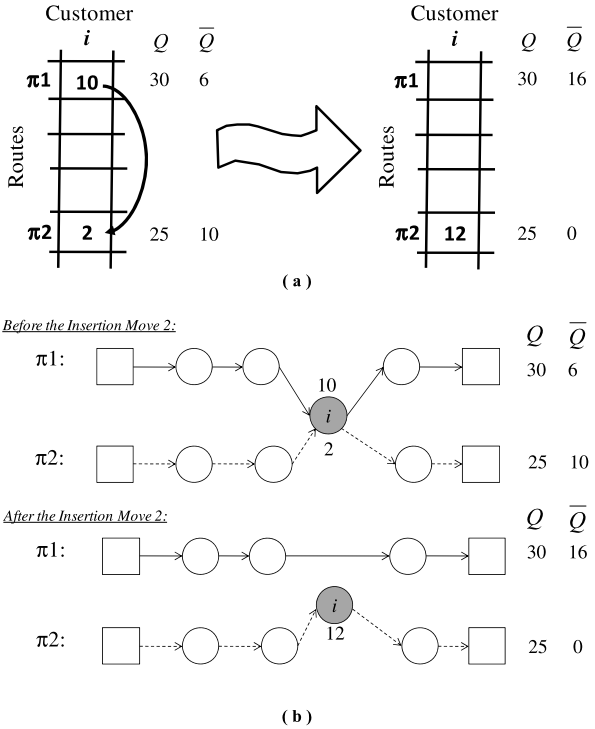

Fig. 5 Insertion move 2. Figure (**a**) illustrates the m_{I2} move—to insert 10 units from route $\pi 1$ to route $\pi 2$ for a given customer i—in the LOAD matrix, whilst figure (**b**) shows its impact on the routes. Note that Q represents total capacity at route π, and $\bar{Q}$ the remaining capacity. Before the move, a customer i is visited by both routes, route $\pi 1$ collects 10 units whereas route $\pi 2$ collects 2 units. After the insertion move, customer i is removed from route $\pi 1$ and route $\pi 2$ collects 12 units

such that i has an assignation $l_{\pi i}$ in $\pi \in \Pi$ and j an assignation $l_{\hat{\pi} j}$ in $\hat{\pi} \in \Pi$, the procedure explores the interchange of $x_{\pi i}^{\hat{\pi} j} = \min\{l_{\pi i}, l_{\hat{\pi} j}\}$ units between both routes. See example in Fig. 6.

3. *Route Combine Move* (m_{C}). This move aims to merge two different routes into a single one, while the routing costs are minimized. For each pair of routes $\pi 1 \neq \pi 2$, the procedure firstly evaluates the union of both routes in that with the largest vehicle. If the union is feasible in terms of the vehicle capacity, and the accessibility and time constraints are satisfied, both routes are merged in a single one. Otherwise the procedure seeks a larger vehicle in the fleet. See example in Fig. 7.
4. *Route Elimination Move* (m_{E}). This move aims to reduce the number of routes, by reallocating the entire load of a route among the others. For each route $\pi \in \Pi$, let C be the set of customers with a load assigned in route π, and $\Pi^i \subseteq \Pi \backslash \{\pi\}$ the subset of feasible, unrestricted and non-penalized routes for customer $i \in C$. While $C \neq \varnothing$ or the reallocation of the entire load of π among the others routes was impossible, repeat: compute the largest portion of load assigned to π that may be reassigned to any other route $\hat{\pi} \in \Pi^i$ as $x_{\pi i}^{\hat{\pi}} = \min\{\bar{Q}_{\hat{\pi}}, l_{\pi i}\}$. Let $i^* = \mathrm{argmax}_i\{x_{\pi i}^{\hat{\pi}}\}$ the customer i who would reassign the largest amount of load from π to any other route $\hat{\pi} \in \Pi^i$. In the case of a tie, the customer i^* that, when inserted into route $\hat{\pi}$ yields the greatest routing cost saving is selected. Update i^*, $x_{\pi i}^{\hat{\pi}}$ and $\hat{\pi}$. If $x_{\pi i}^{\hat{\pi}} = l_{\pi i}$ then customer i^* is deleted from C. If the entire load of a route π was reassigned and the saving obtained by eliminating the route is greater

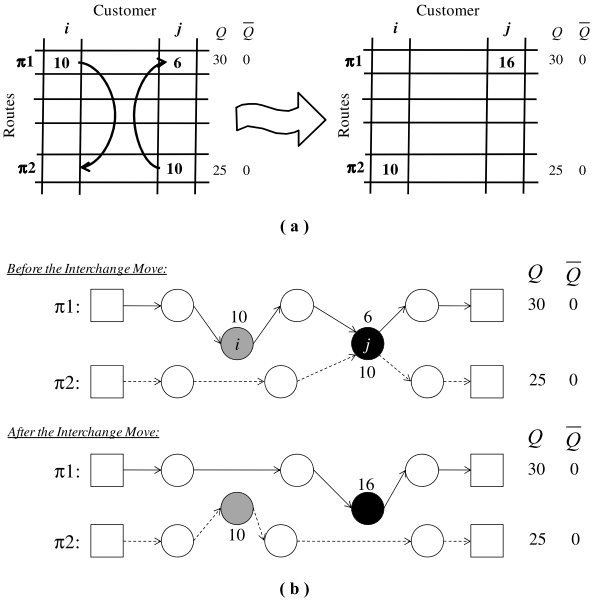

Fig. 6 Interchange move. Figure (**a**) illustrates the m_{E2} move—to interchange 10 units between routes $\pi 1$ and $\pi 2$ for two customers i and j—in the LOAD matrix, whilst figure (**b**) shows its impact on the routes. Note that Q represents the total capacity of route π, and $\bar{Q}$ the remaining capacity. Before the move a customer i is visited by route $\pi 1$ collecting 10 units from it, whereas customer j is visited by both routes, collecting 6 units in route $\pi 1$ and 10 units in route $\pi 2$. After the interchange move, route $\pi 2$ collects 10 units from customer i, and route $\pi 1$ collects 16 units from customer j. Note that the remaining capacity of both routes remains unchanged

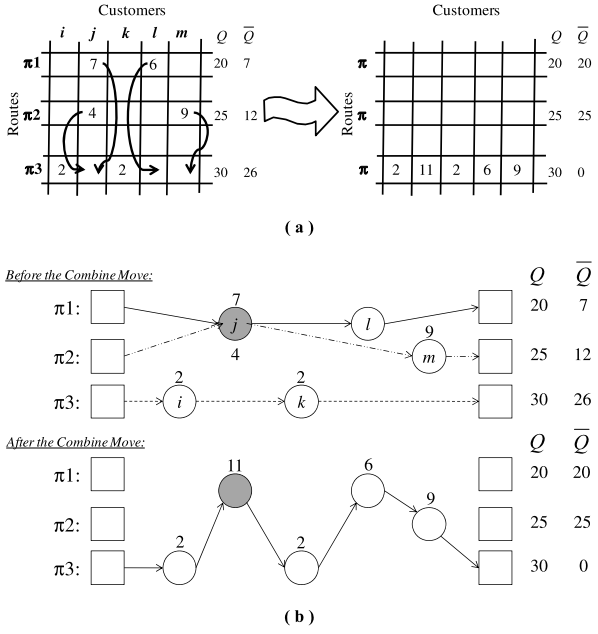

Fig. 7 Combine move. Figure (**a**) illustrates the m_{C} move—to combine routes $\pi 1$ and $\pi 2$ into route $\pi 3$—in the LOAD matrix, whilst figure (**b**) shows its impact on the routes. Note that Q represents total capacity of route π, and $\bar{Q}$ the remaining capacity. Before the move a route $\pi 1$ collects 13 units (7 from customer j and 6 from customer l), whilst a route $\pi 2$ also collects 13 units (4 from customer j and 9 from customer m). Note that the remaining capacity of both routes is not enough to merge them into a single route $\pi 1$ or $\pi 2$. However route $\pi 3$ has enough remaining capacity to visit all customers considered. After the combine move, routes $\pi 1$ and $\pi 2$ are eliminated, whereas all the capacity at route $\pi 3$ is used

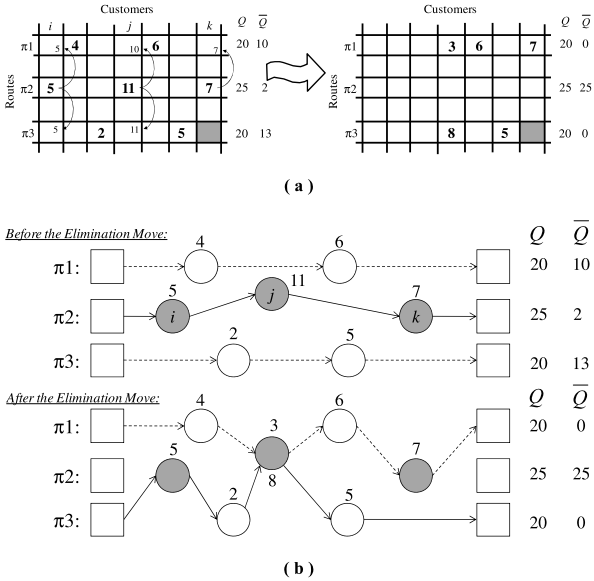

Fig. 8 Elimination move. Figure (**a**) illustrates the m_{E} move—to eliminate route $\pi 2$ by reallocating its entire load between the routes $\pi 1$ and $\pi 3$—whilst figure (**b**) shows its impact on the routes. Note that Q represents total capacity at route π, and $\bar{Q}$ the remaining capacity. Before the move, a route $\pi 2$ collects 23 units (5 from customer i, 11 from customer j and 7 from customer k). After the elimination move, all demand from customer i, and 8 units from customer j are collected by route $\pi 3$, while the remaining 3 units from customer j, and 7 units from customer k are collected by route $\pi 1$. Arrows in figure (**a**) represent all feasible alternatives for each customer, taking into account the remaining capacities Q. Note that route $\pi 3$ is infeasible for customer k

than the reallocations cost, then the movement is recorded as valid. See example in Fig. 8.

4 Empirical work

In this section we present the experimental results obtained with an implementation of the GRASP algorithm for this vehicle routing problem. The heuristic was coded in C using the Microsoft Visual C++ 6.0 programming environment via the Windows Vista operating system. All experiments were conducted on an Acer AMD Athlon X2 PC with a processor operating at 1.90 GHz and 2.00 GB RAM. The proposed algorithm must be capable of efficiently solving real-world sized instances of the problem described above. To test whether this is so, a sufficiently large number of instances is required for analyzing the performance of the algorithm. However, large-sized instances were not available in the literature published to date. Moreover, to the best of our knowledge, the issue of VRP with date windows has not been addressed before. To overcome this drawback, a random instance generator was designed to create instances for which *high quality feasible solutions* can be estimated, allowing us to compare our results.

Although this is not a common practice, this approach has been used in the literature when describing a new problem for which, besides the absence of instances against which to make a benchmark, there is not a mathematical formulation of the problem (see for example González and Adenso-Díaz 2006), or when, although there is a mathematical formulation of the problem, it is of such a great complexity that it is not possible to find bounds by a "brute force" approach using a MIP commercial software (see for example Taillard 1993). In González and Adenso-Díaz (2006) the pseudo-optimal (or best-known solution) of the instances for its disassembly sequence problem is obtained by disassembling the components in the reverse order of their assembly sequence, which is always available. Taillard (1993) tests his proposed algorithms on a set of random generated problems for which he conjectures that the optimum solutions are known. For him the optimality has been conjectured because his algorithm has produced such a type of solution for small instances for which he has not found any evidence for such solutions to not be optimum.

4.1 Instance generator

Real-world instances are mainly characterized by situations in which the date window and the collection capacity are both moderate and highly constrained. However, with the aim of studying the behavior of the algorithm under different situations, we have defined the random instances generator considering the following factors: F1—*number of customers* {F_{11}—50, F_{12}—100, F_{13}—150}; F2—*demand type* {F_{21}—balanced among all the customers, F_{22}—unbalanced}; F3—*date windows tightness* {F_{31}—tightly, F_{32}—halfway, F_{33}—loosely}, F4—*capacity tightness regarding demand* {F_{41}—collection capacity of 15% over the total customers demand, F_{42}—idem 50%, F_{43}—idem 85%}, F5—*dispersion of customers grouped in clusters*, measured as the distance from the center of the cluster to the depot {F_{51}—1/3, F_{52}—1/15}. An instance assignment (or class) is defined by a combination of these five factors. It should be noted that the number of classes obtained is $(3)(2)(3)(3)(2) = 108$.

Due to the complexity of the problem, *optimal solutions* with the minimum routing costs are difficult to obtain. Therefore, the purpose of the instance generator is to create random instances for which *high quality feasible solutions* can be estimated. These best-known feasible solutions will be referred to, from now on, as BK solutions. It should be noted that BK solutions are not obtained from the instance by applying some kind of heuristic procedure (i.e., BK solutions are not *heuristic solutions*). Instead, for each instance, we start by designing a solution and then we derive the instance data such as the distances and travel times for that solution. Thus our GRASP algorithm, without a previous knowledge of the solution, will try to find the solution or even improve it. We describe below the procedure to generate the random instances.

Each instance contains the following data: the number of customers to be served, the number of vehicles (with its corresponding capacities), and the number of trips allowed each vehicle. Thus, the total collection capacity of the systems is computed as: $\gamma = \sum_{k \in K} Q_k \cdot T \cdot R$. The customer's demand is computed and generated considering factors F1, F2 and F4. The next step consists of generating the LOAD matrix (i.e., the solution), which is initialized by taking the first customer in the set and inserting a portion of its demand into the first route where both the largest load may be

allocated and the remaining capacity of the vehicle is minimized. This procedure is repeated until the total customer's demand has been allocated. Then, the data structure is updated and the next customer in a lexicographical order is selected. This step ends when the total demand of all customers has been allocated.

To ensure full truckload routes, the customer's demand is complemented in each route that requires it. To define the date windows, for each customer the procedure identifies the first and last day in which it is visited. Then, via the factor F3 consideration, it determines the width of the date windows. The size of the area to locate customers ranges from $(-50$ to $50)$ for x and y, the depot is located at $(0, 0)$. To ensure *high-quality feasible routes*, to be used as benchmarks, all customers assigned to the same route are grouped into clusters, the coordinates of the clusters are calculated, and their amplitude is defined by the factor F5.

It should be noted that for instances with spread customers this solution could not be close to a high-quality solution, however this solution is always available, and therefore it can be used as a reasonable reference to measure the efficacy of the algorithm. All instances data are available from the authors under request.

4.2 Analysis and discussion of some computational results

The experimental framework consists of three different experimental situations used to analyze the performance of the algorithm proposed and to gain insights into the implementation issues. With this aim, the solution found by GRASP was compared to the BK solution for each instance in its corresponding experiment. To measure the difference between both solutions the percentage deviation (*DEV*) is computed as $\frac{Z_{GRASP}-Z_{BK}}{Z_{BK}} \times 100$, where Z_{GRASP} and Z_{BK} are the GRASP and BK solutions respectively. The remainder of this section describes the design of every experiment and discuses its results.

Experiment A—behavior of the quality parameter α This experiment is focused on the appropriate choice of the quality parameter α in the construction phase, which regulates the size of the Restricted Candidate List. The purpose of this experiment is to evaluate the algorithmic performance as a function of the value of α. To evaluate the performance of the algorithm in terms of α, we compare the results obtained with the different values of $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, on a subset of nine problem classes. A summary of the results is displayed in Table 2. For each problem class the average percent deviation from BK, the worst percent deviation from BK, and the percentage of infeasible solutions in the construction phase are shown. The average is computed over 5 instances from each class. The class of problems selected to perform the analysis corresponds to instances with 100 customers, which are spread $1/3$ of the distance between the center of the cluster to which they are assigned and the depot, and with an unbalanced demand. The variation in the date windows and load capacity characterize each problem class. The choice of problem class is justified because we want to evaluate the algorithmic performance in problems with both date windows and load capacity that are loosely constrained and harder constrained, to resemble the situations currently observed in real-world applications. In addition, performing an Analysis of Variance test it can be demonstrated that factors such as the date windows and load capacity affect the quality of the obtained solutions.

Table 2 Evaluation of quality parameter α. For each problem class it shows: (A) the average percent deviation from BK, (B) the worst percent deviation from BK, and (C) the percent of infeasible solutions in the construction phase

Problem Class		$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.9$
DS1 – ⟨2, 2, 1, 1, 1⟩	(A)	4.69	5.83	−2.72	4.93	4.20
	(B)	7.88	10.69	5.97	10.25	9.79
	(C)	12.80	11.60	13.60	12.40	13.20
DS2 – ⟨2, 2, 1, 2, 1⟩	(A)	3.09	−1.87	0.56	1.12	−4.86
	(B)	4.77	2.47	2.41	3.10	−0.11
	(C)	0.00	0.00	10.23	14.00	67.20
DS3 – ⟨2, 2, 1, 3, 1⟩	(A)	−1.07	−0.27	−0.38	−0.10	−1.93
	(B)	1.31	0.77	0.25	1.17	1.16
	(C)	0.00	0.00	0.00	0.00	0.00
DS4 – ⟨2, 2, 2, 1, 1⟩	(A)	2.32	3.82	3.24	3.74	2.28
	(B)	4.73	6.43	5.10	5.87	5.15
	(C)	0.00	0.00	0.00	0.00	0.00
DS5 – ⟨2, 2, 2, 2, 1⟩	(A)	4.32	5.54	6.19	9.68	7.57
	(B)	16.33	16.56	8.55	17.57	17.58
	(C)	0.00	0.00	0.00	0.00	0.00
DS6 – ⟨2, 2, 2, 3, 1⟩	(A)	0.16	2.23	6.05	8.74	11.68
	(B)	6.10	8.87	9.39	13.71	16.36
	(C)	0.00	0.00	0.00	0.00	0.00
DS7 – ⟨2, 2, 3, 1, 1⟩	(A)	30.69	37.42	33.46	28.91	30.82
	(B)	47.42	44.32	50.88	48.91	47.26
	(C)	0.00	0.00	0.00	0.00	0.00
DS8 – ⟨2, 2, 3, 2, 1⟩	(A)	34.38	30.07	37.34	39.44	37.10
	(B)	47.79	42.96	43.61	45.06	46.45
	(C)	0.00	0.00	0.00	0.00	0.00
DS9 – ⟨2, 2, 3, 3, 1⟩	(A)	25.84	29.42	32.43	35.66	40.85
	(B)	41.55	43.57	34.78	43.77	46.08
	(C)	0.00	0.00	0.00	0.00	0.00

As can be seen, for the instances in the DS1, DS2, and DS3 class (characterized by hardly-constrained date windows), the best results were obtained with $\alpha = 0.5$, whereas for instances in DS4, DS5, and DS6 class, $\alpha = 0.1$ gave the best results. Possibly, $\alpha = 0.4$ is a better compromise in this sense. For the instances in the DS7, DS8, and DS9, $\alpha = 0.1$ gave the best results, even if the worst relative gap is the largest one. In any case, the results obtained indicate that as the instances become more tightly constrained and, thus, more difficult to solve, medium to low values of α can be preferred. From the analysis of the table, it can be noticed that the percent

Table 3 Evaluation of the local search. For each problem class it shows: the average and the worst percent deviation from BEST and the average ratio time required by each phase of our algorithm

Problem class		Construction phase		Local search phase	
		Deviation from BK (%)	Ratio time (%)	Deviation from BK (%)	Ration time (%)
DS1 – ⟨2, 2, 1, 1, 1⟩	Avg.	23.06	17.43	2.92	82.57
	Worst	28.12		4.59	
DS2 – ⟨2, 2, 1, 2, 1⟩	Avg.	25.18	19.11	0.78	80.89
	Worst	32.90		2.00	
DS3 – ⟨2, 2, 1, 3, 1⟩	Avg.	28.74	10.12	0.00	89.88
	Worst	35.76		0.00	
DS4 – ⟨2, 2, 2, 1, 1⟩	Avg.	34.93	17.19	4.33	82.81
	Worst	39.84		7.87	
DS5 – ⟨2, 2, 2, 2, 1⟩	Avg.	29.04	18.37	4.86	81.63
	Worst	39.42		7.83	
DS6 – ⟨2, 2, 2, 3, 1⟩	Avg.	30.48	14.45	4.47	85.55
	Worst	38.98		6.88	
DS7 – ⟨2, 2, 3, 1, 1⟩	Avg.	47.61	14.56	28.90	83.44
	Worst	62.50		39.79	
DS8 – ⟨2, 2, 3, 2, 1⟩	Avg.	50.36	16.70	39.37	83.3
	Worst	59.22		42.13	
DS9 – ⟨2, 2, 3, 3, 1⟩	Avg.	44.82	16.93	33.07	83.07
	Worst	51.40		42.07	

deviation clearly increases as higher α values are used, meaning that in terms of solution quality the variants which favor greedy implementations are to be preferred to random implementations. Moreover, high values of α require a significantly larger computational time, which is especially spent by the local search phase to reach local optima from highly-random solutions.

Experiment B—behavior of the local search procedure The aim of this experiment was to investigate the behavior of the local search procedures. To this end we set the GRASP parameters $iters = 100$ and $\alpha = 0.4$, and we compare the results obtained with the same set of instances used in Experiment A. A summary of the results is displayed in Table 3. For each problem class the average percent deviation obtained after each phase of our GRASP algorithm in comparison with the *best solution known* (BEST) is shown. Such a comparison permits us to evaluate the importance of each phase of our GRASP algorithm, including the local search procedures, in the overall quality of the solutions generated. The time ratio used for each phase in comparison with the total solution time are also shown.

As can be seen, the results in Table 3 indicate that the local search indeed provides considerably improved solutions. For instance, for the DS2 class the average percent deviation from BEST obtained by the greedy randomized construction phase is 25.18%. The local search phase permits an additional improvement, reaching a final solution that is deviated 0.78% on average from BEST. The greedy randomized construction phase takes 19.11% of the total time, while the local search takes the remaining 80.89%. Most of the solutions obtained after the greedy randomized phase are feasible (i.e., non-penalized solutions), but are often far from the best-known solutions (34.91% on average and up to 62.50% in some instances). The local search phase improves these results significantly (32.08% of improvement on average). Even, the instances with poor results (DS7, DS8, and DS9, which correspond to classical VRP instances) show an improvement, though still poor results (with a final solution above 25% from BEST) following the local search phase.

In some GRASP solutions, infeasible solutions were generated. But even in such cases the local search phase was able to reduce infeasibilities. In general, initial solutions were improved by 23.71%. The phase I local search moves, which were able to reduce infeasibilities in all cases, are especially relevant. Moreover, Phase I moves were required 20% of the times, attaining an average improvement of 68.91% over the initial solution. Regarding the phase II moves, the m_{I2} and m_{E2} moves were applied 99.05% and 93.64% of the times respectively, while m_C was applied 65.77% of the times. These improvements have a cost: the local search phase uses the greater part of the CPU time (more than 80%).

Experiment C—solution spectrum of our complete GRASP procedure The aim of this experiment was to analyze the influence of different factors on the quality of the solution obtained and on the efficiency of the algorithm. With this aim, an experimental design was established to include the five factors (from F1 to F5) used in the generation of the random instances. We set the GRASP parameters $iters = 100$ and $\alpha = 0.4$. In order to gain an insight into the performance of the algorithm proposed, the percentage deviation was computed for each instance. The number of observations obtained for the dependent variable DEV is 540, which corresponds to five replicates for each one of the 108 problem classes.

Experimental results show a good performance of the algorithm proposed: on average, the metaheuristic solution obtained is at 4.32% of the BK solution in an average computing time of 113.68 seconds. In fact, for many instances (225) our approach was able to reach better solutions than the BK solution. The average DEV for each combination of factors ranges from −7.49% to 40.37%. This situation tells us that our algorithm performs better in some combinations of factors than in others. Therefore, the results were analyzed using a 5-way ANOVA with repeated measures, where the five factors were considered to have fixed effects and are the object of the analysis.

Table 4 shows the inter-subject effects corresponding to the dependent variable DEV. The last column gives the significance level of each factor (two-way interaction respectively), which measures the probability of occurrence of the observed averages of the dependent variable for the different levels of the factor (combination of levels of two factors respectively) if they all had the same mean. Thus, if the significance level of a certain factor is below the significance level 0.05, this means that we can

Table 4 ANOVA table for the quality of the solution *DEV* ($R^2 = 0.827$)

Source	D. of freedom	Sum of squares	Mean square	F	Significance
F1	2	164386	81743	332.97	0.000
F2	1	438	438	1.78	0.182
F3	2	5179	2590	10.55	0.000
F4	2	354969	177485	722.96	0.000
F5	1	17712	17712	72.15	0.000
F1*F2	2	1905	953	3.88	0.021
F1*F3	4	4317	1079	4.40	0.002
F1*F4	4	21203	5301	21.59	0.000
F1*F5	2	3086	1543	6.29	0.002
F2*F3	2	1564	782	3.18	0.042
F2*F4	2	1345	673	2.74	0.066
F2*F5	1	40	40	0.16	0.686
F3*F4	4	13356	3339	13.60	0.000
F3*F5	2	431	216	0.88	0.416
F4*F5	2	5706	2853	11.62	0.000
Error	506	124221	245		
Total	539	718960			

consider that factor as statistically significant, thus having an influence on the results of our algorithm.

Analyzing the factorial design with the ANOVA, it can be seen that factors F1, F3, F4 and F5 are statistically significant for DEV. It cannot be demonstrated that a factor such as the demand type (F2) affects the quality of the solutions obtained. Since factors F3 and F4 are obviously significant (because each combination level corresponds to a different instance), this result implies that the problem class makes a difference. All two-way interactions are significant at the 0.05 level, except some that involve factor F5. The significance level of the two-way interaction between F2 and F5 is larger than 0.60. This implies that there are not significant interactions between the demand type and the dispersion of customers, as also does the dispersion of customers and the date windows tightness. It should be noted that the model fit ($R^2 = 0.827$) indicates that the five factors and their two-way interactions explain much of the variance of the observed results.

An in-depth analysis reveals that the algorithm performs better in instances in which the date window and the collection capacity are more tightly constrained. On the other hand, the worst results are obtained in those instances in which the collection capacity is larger and the customer demand is smaller. This is because the algorithm was designed to deal with the first group of instances, while instances easier to solve with wide date windows, large collection capacity and customers with a low demand do not require being split. In fact, such instances resemble classical VRP problems, wherein inter and intra routes moves are required to improve heuristic solutions.

Table 5 Input data to do the case study

Depot	Installed capacity (items)	Number of stores served	Range of items collected by store
1	378	166	5–159
2	326	175	5–228
3	387	128	3–168
4	306	136	2–529
5	293	102	4–77

5 Case study

In this part of the work, we perform a comparison of the proposed approach with current practice. To do so, we consider the particular situation of a national WEEE Collective Management System operating in Galicia (the northwest of Spain), which, in 2008, collected more than 3,239 tons of WEEE from 707 geographically dispersed stores via 5 depots. Data related to the fleet of vehicles used in the collection process, distances and travel times, as well as the real amount of items collected throughout 2008 in every store, were provided by the collective management system. In general, two types of vehicles with different load capacities 15 and 25 items) are used in the collecting process. The first (hereafter referred to as vehicle type 1) are smaller than the corresponding vehicles with a load capacity of 25 items (hereinafter referred to as vehicle type 2). They therefore can access to all stores, although the latter have a loading platform that facilitates the collection task. Annual fixed costs are respectively 34,993.47 and 41,723.96 €/year, while the variable costs reach 0.15 and 0.17 €/km. To design the collection routes, for each depot we selected the week with the highest demand collected within the last quarter of 2008 (see Table 5)—given that during this period a significant upturn in WEEE generation was produced due to governmental policies to promote recycling—taking what could be considered a worst-case scenario. It should be noted that each store has its own demand, and its corresponding date window. The GRASP parameters were set on $iters = 100$ and $\alpha = 0.5$.

By applying the heuristic procedure all over the five depots, 229 routes were generated totaling 15,589.62 km. Regarding the tour length characteristics, Table 6 shows that, except for depot 2, the number of routes per depot and the number of kilometers traveled between the corresponding areas remain almost constant. The distance traveled in each depot provides information on two parameters: the size of territory to be covered by the depot, and the number of stores, thus characterizing the differences between the five regions.

To perform the comparison of the approach proposed with current practice, the field data were analyzed to estimate the transportation cost of the current delivery method from stores to the corresponding regional depots. The datasets employed contained only the number of items collected each day from each store for the corresponding regional depot, and they did not specify the types of trucks or the number of trucks that were used. Additionally, the records did not include specific cost information. Hence, some assumptions to estimate the cost were made, reflecting the current standard for routing and scheduling WEEE collection activities. From the records, the number of items was transformed to the type and number of trucks that might be

Table 6 Routing details in each depot, given by the algorithm proposed

Depot	Number of tours	Number of trucks		Kilometers traveled
		Type 1	Type 2	
1	39	1	2	1,677.99
2	63	1	5	4,711.45
3	44	1	2	2,824.23
4	48	1	2	3,142.94
5	35	1	2	3,233.01

Table 7 Comparison between the current method and the GRASP algorithm

Depot	Current operation				Operation proposed	
	No. of tours	Km. traveled	No. of trucks	Cost (€/week)	Cost (€/week)	% of cost reduction
1	73	3,011	4	3,143.31	2,561.60	18.51%
2	118	8,638	5	4,660.41	3,875.47	16.84%
3	84	5,092	4	3,455.55	2,748.13	20.47%
4	86	5,330	4	3,491.34	2,784.25	20.25%
5	66	6,210	3	2,950.37	2,840.05	3.74%

used. We assume that type 1 trucks were used, and based on the analysis of current practices, this assumption appears valid. We assume that, whenever possible, a full truck load was used, and we also attempted to minimize the number of trucks used. In other words, routes are optimized based on the known supply of end-of-life electronic items from each store. After the number of tours was estimated, we calculated the distances of the entire tours by multiplying the two-way distance from each store to the corresponding regional depot. To compare the performance of the two methods more directly, we estimated the transportation cost of each method. The results are presented in Table 7, along with the results of the algorithm proposed.

Even though one cannot directly compare the distances of the two methods because of the different types of computation used, distance was reduced to between 1,333 and 3,927 km with the method proposed, and the number of tours was reduced to between 31 and 55. The transportation cost of the five depots fell an average of 15.96% from the current levels per week, with the largest reduction at depot 3, which had the largest installed capacity, and the smallest reduction occurred at depot 5, which had the smaller territory. Results indicate that both the appropriate use of vehicles and a suitable splitting of loads is a determinant factor in the performance of any WEEE collection system.

6 Conclusions

The Vehicle Routing Problem with Split Loads and Date Windows is a problem of practical relevance. Due to the computational complexity of the problem, it is important to develop polynomial time heuristic solution procedures. In this the VRPSLDW

was formulated as a Mixed Integer Programming model, and a GRASP-based metaheuristic algorithm was designed to solve it. Computational results indicate that the proposed algorithm satisfactorily handles a real-life problem as well as other random instances generated. Experimental results on a set of 540 random instances generated show the convenience of the algorithm to solve problems in which the date window and the collection capacity are tightly constrained. This kind of problems are commonly found in real situations. As a result, a good behavior in real problems is expected. The performance of the metaheuristic algorithm was analyzed with respect to the amount of processing time required by the procedure to generate a solution and the quality of the solutions.

The results of the ANOVA test showed that factors like the date windows and the capacity tightness regarding demand are statistically significant for the algorithm performance. On the other hand, it cannot be demonstrated that a factor like the dispersion of the customers affects the quality of the solutions obtained, which is a sign of the robustness of the algorithm to deal with this kind of problems. Results from the local search analysis indicate that the local search indeed provides considerably improved solutions. Especially relevant are the phase I local search moves which were able to reduce infeasibilities in all cases. Finally, the comparison of the proposed approach with current practice reveals that the transportation cost of the five depots falls an average of 15.96% from the current levels.

As a future work it would be desirable to derive lower bounds for the VRPSLDW to further evaluate the results obtainable. Modifications of existing exact solution procedures for the split delivery vehicle routing problem could yield such bounds. Also, there are many ways in which our proposed algorithm can be improved so that the number and length of tours needed to reach quality levels can be reduced, thus making its application to larger problem instances feasible. First, local optimization heuristics like 2-opt, 3-opt, GENUIS or Lin-Kernighan can be embedded in the algorithm (this is a standard approach to improve efficiency of general purpose algorithms for vehicle routing problems) to improve the performance of long routes. Second, the development of alternative metaheuristics algorithms, such as Tabu Search or Scatter Search, which have proved to give good results in classical vehicle routing problems, could be considered.

Acknowledgements This research has been partially funded by the Spanish Ministry of Science and Innovation grant MEC-DPI2010-16201 and FEDER, by the Mexican National Council for Science and Technology (Grant SEP-CONACYT 130053), and by Tecnológico de Monterrey Research Fund CAT128. We are grateful to two anonymous reviewers whose comments have helped to improve the presentation of this work.

References

Aleman, R.A., Zhang, X., Hill, R.R.: An adaptive memory algorithm for the split delivery vehicle routing problem. J. Heuristics **16**, 441–473 (2010)

Alonso, F., Alvarez, M.J., Beasley, J.E.: A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. J. Oper. Res. Soc. **59**, 963–976 (2007)

Álvarez, A., Pacheco, J., Angel-Bello, F., Garcia, I.: Un problema de distribución de productos alimenticios con flexibilidad en la fecha de entrega. In: VI Congreso de Metaheuristicos, Algoritmos Evolutivos y Bioinspirados, Spain (2009)

Battarra, M., Monaci, M., Vigo, D.: An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. Comput. Oper. Res. **36**, 3041–3050 (2009)
Belfiore, P., Yoshida-Yoshizaki, H.T.: Scatter search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in Brazil. Eur. J. Oper. Res. **199**, 750–758 (2008)
Beullens, P., Van Wassenhove, L., Van Oudheusden, D.: Collection and vehicle routing issues in reverse logistics. In: Dekker, Fleischmann, Inderfurth, Van Wassenhove (eds.) Reverse Logistics, Quantitative Models for Closed-Loop Supply Chains 2004, pp. 275–291. Springer, Dordrecht (2004)
Blanc, I., Van Krieken, M., Krikke, H., Fleuren, H.: Vehicle routing concepts in the closed-loop container network of ARN—a case study. OR Spektrum **28**, 53–71 (2006)
Brandão, J., Mercer, A.: The multi-trip vehicle routing problem. J. Oper. Res. Soc. **49**, 799–805 (1998)
Dror, M., Trudeau, P.: Savings by split delivery routing. Transp. Sci. **23**, 141–145 (1989)
Eksioglu, B., Volkan-Vural, A., Reisman, A.: The vehicle routing problem: a taxonomic review. Comput. Ind. Eng. **57**, 1472–1483 (2009)
European Parliament the Council and the Commission: Directive 2002/96/EC of 27 January 2003 on waste electrical and electronic equipment (WEEE). Off. J. Eur. Union **L37**, 24–39 (2003). http://www2.uca.es/grup-invest/cit/Union%20Europea_archivos/WEEE_ingles.pdf. Accessed 15 January 2009
Feo, T.A., Resende, M.G.C.: A probabilistic heuristic for a computationally difficult set covering problem. Oper. Res. Lett. **8**, 67–71 (1989)
Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. J. Glob. Optim. **6**, 109–133 (1995)
Golden, B.L., Assad, A., Levy, L., Gheysens, F.: The fleet size and mix vehicle routing problem. Comput. Oper. Res. **11**, 49–65 (1984)
González, B., Adenso-Díaz, B.: A scatter search approach to the optimum disassembly sequence problem. Comput. Oper. Res. **33**, 1776–1793 (2006)
Imran, A., Salhi, S., Wassan, N.: A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem. Eur. J. Oper. Res. **197**, 509–518 (2009)
Kim, H., Jaehwan, J., Lee, K.D.: Vehicle routing in reverse logistics for recycling end-of-life consumer electronic goods in South Korea. Transp. Res. D **14**, 291–299 (2009)
Mar-Ortiz, J., Adenso-Díaz, B., González-Velarde, J.L.: Design of a recovery network for WEEE collection: the case of Galicia, Spain. J. Oper. Res. Soc. (2010). doi:10.1057/jors.2010.114
Mitra, S.: A parallel clustering technique for the vehicle routing problem with split deliveries and pickups. J. Oper. Res. Soc. **59**, 1532–1546 (2008)
Nag, B., Golden, B.L., Assad, A.: Vehicle routing with site dependencies. In: Golden, B.L., Assad, A. (eds.) Vehicle Routing: Methods and Studies. Studies in Management Science and Systems, vol. 16, pp. 149–159. North-Holland, Amsterdam (1988)
Schultmann, F., Zumkeller, M., Rentz, O.: Modeling reverse logistic tasks within closed-loop supply chains: An example from the automotive industry. Eur. J. Oper. Res. **171**, 1033–1050 (2006)
Taillard, E.: Parallel iterative search methods for vehicle routing problems. Networks **23**, 661–673 (1993)
Taillard, E., Laporte, G., Gendreau, M.: Vehicle routing problem with multiple use of vehicles. J. Oper. Res. Soc. **47**, 1065–1070 (1996)