Engineering Applications of Artificial Intelligence 42 (2015) 45–56

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

GVNS for a real-world Rich Vehicle Routing Problem with Time Windows

Jesica de Armas*, Belén Melián-Batista, José A. Moreno-Pérez, Julio Brito

Universidad de La Laguna, Dpto. de Ingeniería Informática y Sistemas, 38271 La Laguna, Spain

ARTICLE INFO

Article history:
Received 12 September 2014
Received in revised form
15 January 2015
Accepted 19 March 2015
Available online 10 April 2015

Keywords:
Rich VRPTW
Fixed Heterogeneous Fleet
General Variable Neighbourhood Search
Metaheuristics

ABSTRACT

Rich Vehicle Routing Problems are vehicle routing problems (VRPs) that deal with additional constraints, which aim to better take into account the particularities of real-world applications. They combine multiple attributes, which constitute a complement to the traditional models. This work proposes an adaptive solution method based on metaheuristics for solving a Rich Vehicle Routing Problem with Time Windows. This software has been embedded into the fleet management system of a company in the Canary Islands. The attributes considered by the company are a fixed heterogeneous fleet of vehicles, soft and multiple time windows, customer priorities and vehicle–customer constraints. Furthermore, the company requires the consideration of several objective functions that include travelled distance and time/distance balance. Exact algorithms are not applicable when solving real-life large VRP instances. This work presents a General Variable Neighbourhood Search metaheuristic, which obtains high quality solutions. The computational experiments are presented in four sections, which comprise the parameter setting, the analysis of the effect of the considered attributes, the comparative with the literature for the standard VRP with Time Windows, and the study of the solutions provided by the algorithm when compared with the solutions implemented by the company.

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 requiring a specified volume of goods to be delivered. A fleet of homogeneous vehicles dispatched from a single depot is used to deliver the goods, 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. The VRP has been the subject of intensive research since the 1960s. A wide range of exact methods, heuristics and metaheuristics has been proposed in the specialized literature. We refer the interested reader to the following surveys by Schmid et al. (2013), Vidal et al. (2013), Eksioglu et al. (2009), Potvin (2009), Laporte (2009), Gendreau et al. (2008), Baldacci et al. (2007), and books by Toth and Vigo (2002) and Goldenet al. (2008).

The aim of this work is to solve a real-world VRP that has been posed to the authors by a company in the Canary Islands, Spain. The resulting software has been embedded into a fleet management system. The requirements provided by the company lead to the consideration of several constraints, which have to be integrated into the standard VRP. In the literature, there is a tremendous number of research papers related to VRP with additional constraints, which range from the need of time windows to regulations related to long-distance transportation. With the purpose of collecting all these possible constraints, Vidal et al. (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*). These MAVRPs 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, obtaining the so-called *Rich VRPs* (*RVRPs*) (Schmid et al., 2013; Tarantilis et al., 2009). The problem tackled in this work corresponds to this last class of RVRPs. We refer the interested reader to the taxonomy and definition of Rich VRP proposed by Lahyani et al. (2015).

Due to the difficulty for solving VRPs to optimality, heuristics and metaheuristics constitute an increasingly active research area in the literature. In our work, a General Variable Neighbourhood Search (VNS) algorithm (Hansen et al., 2010b) is proposed for solving a Rich Vehicle Routing Problem with Time Windows (RVRPTW).

The main contributions of this paper relies upon the fact that a Rich VRPTW including several real-world constraints required by some companies has been tackled. First of all, a fixed heterogeneous

* Corresponding author: Universidad de La Laguna, Dpto. de Ingeniería Informática y Sistemas, 38271 la Laguna, Spain.
E-mail addresses: jdearmas@ull.es (J. de Armas), mbmelian@ull.edu.es (B. Melián-Batista), jamoreno@ull.es (J.A. Moreno-Pérez), jbrito@ull.es (J. Brito).

http://dx.doi.org/10.1016/j.engappai.2015.03.009
0952-1976/

fleet of vehicles is considered. Moreover, since 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; allowing the drivers work after their working shift or maximizing the number of customers served postponing the remainder. The problem combines constraints which have not been managed all together in the literature as far as we know. Computational experiments on instances based on the real data and standard benchmark instances have been carried out in this paper.

It is worth mentioning that the solution method proposed in this work, implemented as a metaheuristic, has already been integrated into the optimization tool of the fleet management system used by some companies. The fleet of a company which wants to use this optimization tool must have the necessary devices to communicate with the management system, and then the system will be able to use the optimization tool and provide an optimized route plan. The optimization tool has been implemented using C# and the current solution method has been implemented using C++. Therefore, in order to integrate the metaheuristic method in the optimization tool, a DLL has been developed. The data-interchange format used to communicate the optimization tool with the DLL has been JSON, that is a text-based open standard designed for human-readable data interchange. Furthermore, it is worth noting that through the system interface, the company can activate or deactivate the consideration of the different attributes.

Since the main goal of our work has been to embed the developed software into a commercial fleet management tool, it is desirable to develop an algorithm that not only performs well over the standard VRPTW instances, but also over constrained real-world problems. Therefore, this paper is not aimed at overcoming the best standard VRPTW results, but rather at proposing an algorithm that works effectively for solving real-world instances.

The rest of the paper is organized as follows. Section 2 is devoted to describe the Rich Vehicle Routing Problem with Time Windows (*RVRPTW*) tackled in this work. Section 3 thoroughly describes the General Variable Neighbourhood Search (GVNS) algorithm developed to solve the problem at hand. Section 4 summarizes the computational results carried out over both instances from the literature and real-world data. Finally, the conclusions and future work are reported in Section 5.

2. Rich Vehicle Routing Problem with Time Windows

Vidal et al. (2013) distinguish three main classes of attributes that appear frequently in the literature. These classes are the assignment of customers and routes to resources, the sequence choices, and the evaluation of fixed sequences. The attributes that are taken into consideration in this work are summarized in the following items.

- *Heterogeneous fleet*: In the particular application of VRP tackled in this paper, it is considered a heterogeneous fleet of vehicles. 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 different version of the problem, called Heterogeneous Fleet VRP (*HFVRP*), is revealed. This last problem corresponds to the real-world application solved in this work. Precisely, it is available a fixed set of heterogeneous vehicles. Exact and heuristic methods have been proposed for solving HFVRPs (Baldacci, 2008; Baldacci and Mingozzi, 2009; Brandao, 2011; Li et al., 2007; Paraskevopoulos et al., 2008; Penna et al., 2011; Prins, 2009; Subramanian et al., 2012; Taillard, 1999). 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 of vehicles. Therefore, it might not be possible to obtain a feasible solution for a certain instance. In that case, it is required to provide a valid solution for the company by adding more vehicles, letting the drivers work after their working shift, postponing customers and maximizing the number of served customers, etc.
- *Time windows*: Additional constraints arise if time windows are associated to the depot and customers, obtaining the Vehicle Routing Problem with Time Windows (*VRPTW*). VRPTW and a vast set of its variants have been widely studied in the literature. For recent reviews, see Braysy and Gendreau (2005) and Gendreau et al. (2010). Furthermore, in the practical application reported in this paper, working shifts of vehicles are considered as time windows associated with each vehicle.
- *Soft and multiple time windows*: In the implementation carried out in this paper, additional time attributes, which are the existence of multiple time windows for customers and multiple time intervals in the working shifts for vehicles, are taken into consideration (Ibaraki et al., 2005, 2008). Note that time windows may differ among customers, and working shifts may differ among vehicles. In any case, the customers have to be visited at maximum once during the day. Moreover, soft time windows and soft working shifts are considered, since some of them can be violated, incurring in additional costs. Particularly, if working shifts of vehicles can be extended, extra hours are allowed for the drivers. This leads to additional salary costs, since the extra time is more expensive. A work related to VRP with soft time windows is by Taillard et al. (1997).
- *Customer priority*: In addition to the previous attributes, which are thoroughly analysed in the paper by Vidal et al. (2013), the company under consideration in this work assigns priorities to some customers. Depending on these priorities, some customers can be postponed until the next day and their service is not required during the current planning horizon. Together with extending the working shifts of the vehicles, postponing customers allow the system to obtain valid solutions for the company. 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 are possible alternatives if they are permitted by the company.
- *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 of vehicle–customer constraints that can be due to several reasons such as road restrictions.

In addition to these attributes, *several objective functions* are required by the company to solve the problem at hand. Although the optimality criterion of minimizing the total travelled distance is the most commonly used in the VRP literature, more recent approaches recognize the VRP as a multi-objective optimization problem. Jozefowiez et al. (2008) provide an overview of the research into routing problems with several objectives. Important objectives, besides the minimization of the total travelled distance, are the minimization of the number of vehicles in use, the minimization of the total required time, the maximization of the collected profit and some other objectives related to reaching a balance between the routes. In order to establish a balancing objective, the workload for a route has to be defined. It can be expressed, for example, by the number of visited customers, the cargo, the quantity of delivered goods, the route length or the required time (Borgulya, 2008; Jozefowiez et al., 2008; Kritikos and Ioannou, 2010). Among the researchers who have worked on VRPs considering several objectives and time windows we can mention,

for instance, Hong and Park (1999), who consider the minimization of total vehicle travel time and the minimization of total customer waiting time. Rahoual et al. (2001) consider objectives related to the minimization of the number of used vehicles and the minimization of the total covered distance. More recently, Calvete et al. (2007) minimize the total operational cost, the under-utilization of labour and the vehicle capacity. Jozefowiez et al. (2007) discuss the motivations for applying multi-objective optimization on vehicle routing problems and the potential benefits of doing it. Ghoseiri and Ghannadpour (2010) present a model and a solution method based on genetic algorithms to solve the multi-objective problem considering as objectives both the total required fleet size and total travelling distance. Melián-Batista et al. (2014) consider the objective of balancing routes regarding time in conjunction with time windows in a multi-objective context.

In this work, given the fact that the fleet of vehicles is fixed, the company might require either minimizing the total distance or balancing time or distance if the use of all the available vehicles is mandatory. Therefore, the company has to indicate which principal objective function will be required. If having idle available vehicles is not allowed, then the time/distance balance objective function will be selected as principal one. Time balance is measured as the difference between the longest and shortest routes regarding time. Distance balance is also measured as the difference between the longest and shortest routes regarding distance. Otherwise, minimizing the total distance will be the principal objective function. Furthermore, a set of other objective functions are considered together with the principal one, as explained in the next section; particularly, minimizing the number of vehicles, extra hours, postponed customers and cost. All these functions will be used following a different lexicographic ordering of them, depending on the particular goal in each case.

Finally, *infeasible solutions* are taken into consideration. Particularly, infeasibilities due to the use of more vehicles than available, the extension of the time windows of the customers and the working shifts of the vehicles, or the postponing of customers are tackled.

With the purpose of solving the real-world RVRPTW tackled in this work, we have designed a General Variable Neighbourhood Search algorithm. A tremendous amount of work in the field of vehicle routing problem using VNS has been published. Bräysy (2003) gives the internal design of the Variable Neighbourhood Descent (VND) and Reduced Variable Neighbourhood Search (RVNS) algorithms in detail, analyses the VRPTW problem, and indicates the VND algorithm as one of the most effective ways to solve VRPTW problems. Polacek et al. (2004) design a VNS to solve the multidepot vehicle routing problem with time windows (MDVRPTW). Kytöjoki et al. (2007) design a guided VNS algorithm to handle the 32 existing large-scale VRP problems and compare it with a tabu search (TS) algorithm. Goel and Gruhn (2008) introduce a RVNS to solve the general VRP including time windows, vehicle constraints, path constraints, travel departure time constraints, capacity constraints, order models compatibility constraints, multisupplier point of the orders, and transport and service position constraints. Hemmelmayr et al. (2009) propose a VNS algorithm for periodical VRP. Fleszar et al. (2008) adopt a VNS algorithm to solve the open-loop VRP 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.

3. General Variable Neighbourhood Search for the RVRPTW

Exact algorithmic methodologies are not applicable when solving real-life large vehicle routing problem instances. Therefore, our interest is focused on metaheuristic methodologies that are capable of producing applicable high quality solutions within reasonable computing times. With the purpose of obtaining high quality solutions for the real-world problem at hand, this work proposes an algorithm based on General Variable Neighbourhood Search (GVNS) (Hansen et al., 2010b). Variable Neighbourhood Search (VNS) is a metaheuristic for solving combinatorial and global optimization problems based on a simple principle; systematic changes of neighbourhoods within the search. Many extensions have been made, mainly to be able to solve large problem instances (Hansen et al., 2010a, 2008; Hoeller et al., 2008; Melian, 2006; Moreno-Vega and Melian, 2008).

Let $\mathcal{N}_k$ $(k=1,\ldots,k_{max})$ be a finite set of neighbourhood structures, and $\mathcal{N}_k(s)$ the set of solutions in the kth neighbourhood of a solution s. Usually, a series of nested neighbourhoods is obtained from a single neighbourhood 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 neighbourhood is performed by repeating k times a move into the original neighbourhood. 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). In the implementation performed in this work, the neighbourhoods selected for the shaking process of the VNS are not nested, and different kinds of movements are implemented following the ideas described by Repoussis et al. (2008). The proposed sequence of movements ($k_{max}=6$) 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 neighbourhood structures. The *GENI* operator (Gendreau et al., 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 consecutive customers of a route. The *CROSS* operator (Taillard et al., 1997) selects a sequence of customers from a route, other sequence of customers from other route, and interchanges both sequences. The *2-opt* operator (Croes, 1958) chooses two customers of a route and inverts the sequence of customer visited between them. The *relocate* operator (Cassani and Righini, 2004)deletes a customer from a route and inserts it into another route. The *swapInter* operator selects a customer from a route, other customer from other route, and swaps them.

Additionally, let $\mathcal{N}_l$ $(l=1,\ldots,l_{max})$ be the finite set of neighbourhood structures that will be used in the local search conducted by a Variable Neighbourhood Descent (VND). The Variable Neighbourhood Descent (VND) method is obtained if the change of neighbourhoods is performed in a deterministic way. Its steps are presented in Algorithm 1. The sequence of movements considered in this work ($l_{max}=3$) is the following: *relocate*, *swapIntra* and *swapInter*. In a *VND* algorithm, if solution s' is worst than s, then $l++$; otherwise, $l=1$.

Algorithm 1. Variable Neighbourhood Descent (VND).

```
// Function VND(s,l_max).
1 while (improvement is obtained) do
2 |  Set l ← 1
3 |  while(l < = l_max) do
4 |  |  s' ← argmin_{y ∈ N_l(s)} f(s)
5 |  |  NeighbourhoodChange(s, s', l); //Change neighbourhood
```

In order to solve the RVRPTW, we propose the GVNS metaheuristic, whose pseudocode is shown in Algorithm 2. Once defined the neighbourhood structures $\mathcal{N}_k$ and $\mathcal{N}_l$ in line 1, the best solution is initialized at the empty set and the stopping condition is chosen in lines 2 and 3, respectively. The stopping condition consists of a number of iterations that corresponds to a parameter N that is set in the computational experience. Then, for each iteration, an initial

solution is generated in line 5. With this purpose, 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 larger capacity are selected earlier. 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 earlier. Once having the order of selection of 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 two 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 (1987) algorithm, 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, a new one is created.

Algorithm 2. General Variable Neighbourhood Search (GVNS).

Initialization .
1 Select the set of structures $\mathcal{N}_k$, for $k=1,\ldots,k_{max}$, that will be used in the shaking phase, and the set of neighbourhood structures $\mathcal{N}_l$ for $l=1,\ldots,l_{max}$ that will be used in the local search.
2 Initialize $BestSol \leftarrow \varnothing$.
3 Choose a stopping condition.
4 **While** (*the stopping condition is not met* (*N is not reached*)) **do**
  Generate an initial solution s.
5 // *Iterations*.
6 **while** (*the stopping condition is not met* (*M is not reached*)) **do**
7 (1) Set $k \leftarrow 1$;
8 (2) Repeat the following steps until $k=k_{max}$:
9 (a) *Shaking*. Generate a point s' at random from the kth neighbourhood of s ($s' \in \mathcal{N}_k(s)$).
10 (b) *Local search by VND*.
11 (b1) Set $l \leftarrow 1$;
12 (b2) Repeat the following steps until $l=l_{max}$:
13 $-$ *Exploration of neighbourhood*. Find the best neighbour s'' of s' in $N_l(s')$;
14 $-$ *Move or not*. If $f(s'')<f(s')$, set $s' \leftarrow s''$ and $l \leftarrow 1$; otherwise, set $l \leftarrow l+1$
15 (c) *Move or not*. If this local optimum is better than the incumbent, move there ($s \leftarrow s''$), and continue the search with
16 $\mathcal{N}_1(k \leftarrow 1)$; otherwise, set $k \leftarrow k+1$.
  Update *BestSol*.

The process of creating a new initial solution takes into account several aspects. In the first place, it is worth mentioning the fact that some customers cannot be assigned to certain vehicles due to restrictions. Moreover, if the implemented Solomon heuristic requires more vehicles than available, fictitious vehicles are generated. These vehicles are used to create the other necessary routes and their working shifts are set at the least restrictive values of all initial vehicles. Fictitious vehicles are also included when the working shifts of the remaining vehicles are too restrictive to serve the customers or when the customers are not compatible with the remaining vehicles. Before introducing fictitious vehicles in the case in which compatible vehicles are still available, it is checked if it is allowed to expand their working shifts obtaining extra working hours. If this is permitted, some customers can then be assigned to the current expanded route.

Before continuing the explanation of the procedure proposed in this work, it is worth mentioning that even though several objective functions are considered, the company has to indicate which is the principal objective in each case. Therefore, the total distance, time balance or distance balance can be considered as principal objective. In the last two cases, if the solution obtained with the procedure in line 5 uses less vehicles than available, then an empty route for each unused vehicle will be included in the solution. The rationale behind this is that the company does not want to have any idle available vehicle.

The loop corresponding to lines 6–15 is performed for a number of iterations, M, set by the computational experience. As indicated above, the sequence of neighbourhoods used to carry out the shaking process in GVNS is the following: GENI, Or-Opt, Cross, 2-Opt, Relocate and swapInter. Therefore, line 7 indicates that the first considered neighbourhood is GENI.

The particular implementation performed in this work of these neighbourhoods is summarized in the following items:

- *GENI* : In order to make this movement, the next steps are repeated a certain number of times. Firstly, a source route must be selected among the fictitious routes, but if there is not any fictitious route, a non-empty source route is randomly selected. Then, a destination route must be selected among the empty routes, but if there is not any, the destination route is randomly selected among the three routes which have the closest centroid to the source route. This destination route cannot be fictitious and must have more than one customer. Then, a customer that can be deleted from the source route is selected among the three customers closest to the destination route (sum of distances from this customer to all destination route customers), and the two customers from the destination route closest to the previous one are chosen. If some of these customers cannot be found, the process is tried again. Otherwise, the customer from the source route is inserted between the two customers in the destination route. If the resulting route is infeasible, the movement is undone and the process is tried again. If not, we eliminate the source route from the planning if it is left empty and the main objective in optimization is to minimize the distance or the route is fictitious.
- *Or-opt*: First of all, in order to perform this movement, it is necessary to check if all routes have only two customers. In this case, this movement cannot be carried out. Otherwise, the next steps are repeated a certain number of times. Initially, a route

with more than two customers is randomly selected. Then, two different customers are randomly chosen from this route and a position inside the route is selected to move this sequence of customers. At this point, the movement is done. If the resulting route is infeasible, the movement is undone and the process is tried again.

- *CROSS*: In order to perform this movement, the next steps are repeated a certain number of times. Firstly, a source route with more than one customer is randomly selected. Secondly, a destination route with more than one customer is randomly selected among the three ones which have the closest centroid to the source route. Then, two customers from the source route and two customers from the destination route are randomly selected and the interchange is done. If any of the routes is not feasible, the movement is undone and the process is tried again.
- 2-*opt*: First of all, in order to perform this movement, it is necessary to check if all routes have only one customer. In this case, this movement cannot be carried out. Otherwise, the next steps are repeated a certain number of times. Initially, a route with more than one customer is randomly selected. Then, two different customers are randomly chosen from this route and the sequence is reversed. If the resulting route is infeasible, the movement is undone and the process is tried again.
- *Relocate*: In order to perform this movement, the next steps are repeated a certain number of times. Firstly, a source route must be selected among the fictitious ones, but if there is not any fictitious route, a non-empty source route is randomly selected. Then, a destination route must be selected among the empty ones, but if there is not any empty route, the destination route is randomly selected among the three ones which have the closest centroid to the source route. This destination route cannot be fictitious. Later, a customer which can be deleted from the source route is selected among the three customers closest to the destination route (sum of distances from this customer to all destination route customers), and a customer from the destination route after which the previous one can be feasibly introduced is chosen. If some of these customers cannot be found, the process is tried again. Otherwise, the relocation is done, and we eliminate the source route from the planning if it is left empty and the main objective in optimization is to minimize the distance or the route is fictitious.
- *SwapInter*: In order to perform this movement, the next steps are repeated a certain number of times. Initially, a non-empty first route is selected. Then, a second route must be selected among the three ones that have the closest centroid to the first route. This second route cannot be empty. Then, a customer that can be deleted from the first route is selected among the three closest customers to the destination route (sum of distances from this customer to all second route customers), and a customer that can be deleted from the second route is selected among the three closest customers to the first route (sum of distances from this customer to all first route customers). If some of these customers cannot be found, the process is tried again. Otherwise, the swap is done and its feasibility is checked. If any of the routes is infeasible, the movement is undone and the process is tried again.
- *SwapIntra*: First of all, in order to perform this movement, it is necessary to check if all routes have only one customer. In this case, this movement cannot be carried out. Otherwise, the next steps are repeated a certain number of times. Initially, a route with more than one customer is randomly selected. Then, two different customers are randomly chosen from this route and they are swapped. If the resulting route is infeasible, the movement is undone and the process is tried again.

The processes of shaking, local search and move decision in lines 9, 10 and 15, respectively, are iteratively performed until $k = k_{max}$. First of all, the shaking step in GVNS generates a solution s' at random from the kth neighbourhood 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 $\mathcal{N}_l$ neighbourhoods, which in the implementation proposed for solving the RVRPTW consists of the next sequence of random movements: Relocate, swapIntra and swapInter.

As indicated above, the user of the system has to indicate which the principal objective function will be among minimizing the total distance, time balance or distance balance. In the last two cases, it is considered the difference between the largest and shortest time/distance required by the used vehicles. Additionally, the developed GVNS takes into consideration additional objective functions that have to be minimized using a hierarchical approach. Hierarchic evaluation means that the objective functions are considered in a certain lexicographic order, so that if two selected solutions have equal objective function values for a function, then the next one in the order is considered to break ties. This approach, considered within VNS, has been referred as Variable Formulation Search in the paper by Pardo et al. (2013). In the case in which the total distance is the principal objective, the lexicographic order of the objectives to be minimized is the following.

- Number of fictitious routes.
- Total travelled distance.
- Total number of routes.
- Time balance.
- Salary costs incurred for expanding the working shifts of the vehicles (extra hours are more expensive).

The case in which the principal objective function is either time or distance balance, the objective functions order is the following. The objective functions have to be minimized.

- Number of fictitious routes.
- Time/distance balance.
- Total travelled distance.
- Salary costs incurred for expanding the working shifts of the vehicles (extra hours are more expensive).

Note that in this case, the whole fleet of vehicles is required to be used as indicated by the company. Therefore, minimizing the number of routes is not an objective to be considered.

The rationale behind considering the number of fictitious routes as the first objective function to be minimized is the fact that they can lead to undesirable infeasible solutions. If the customer services assigned to fictitious routes cannot be relocated in other routes and the company does not allow postponing them to the next day, then an infeasible solution is obtained. This result can also be valid for the company since, in this case, it can rent additional vehicles for a single day. In any case, a feasible solution is always preferred.

After one of the N iterations, a solution to the problem s, either feasible or infeasible, is obtained. Since several iterations are carried out to finally select the best alternative solution for the company, in line 16, the best solution is updated. Two different goals that substitute the minimization of the number of fictitious routes are taken into account. The obtained lexicographic order corresponding to the travelled distance as principal objective is the following.

- Number of postponed services.
- Number of extra hours.
- Total travelled distance.
- Number of routes.
- Time balance.

- Salary costs incurred for expanding the working shifts of the vehicles (extra hours are more expensive).

The obtained lexicographic order corresponding to the time/distance balance as principal objective is the following.

- Number of postponed services.
- Number of extra hours.
- Time/distance balance.
- Total travelled distance.
- Salary costs incurred for expanding the working shifts of the vehicles (extra hours are more expensive).

In other words, after one of the N iterations, a solution to the problem s is obtained. It is then compared with the best solution reached by the algorithm so far stored in *BestSol*. If in *BestSol* there are unserved customers or extra hours for the vehicles, but in the new solution s there are not, s is the new *BestSol*. If neither *BestSol* nor s has unserved customers or extra hours, the new *BestSol* is the solution with the least total travelled distance. If *BestSol* does not have unserved customers, but s does, we keep *BestSol*. Otherwise, *BestSol* is updated by s. If *BestSol* does not require extra hours, but s does, we keep *BestSol*. Otherwise, *BestSol* is updated by s.

Therefore, in order to update the best solution in line 16, it is given higher priority to those solutions that include all the services and adjust to the working shifts of the drivers. These two aspects are not considered while running the GVNS, since the extra hours remain unchanged, on one hand, and determining the services belonging to fictitious routes that can be postponed requires high computational costs, on the other hand.

To summarize the description of the proposed algorithm, it is noteworthy that in order to carry out the optimization process, the system requires that the user specifies the desirable principal objective function (distance, time balance or distance balance), if extra hours are permitted and what service priorities allow postponing services.

4. Computational experiments

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 the RVRPTW. Our algorithm has been coded in C++ and runs on a machine with Intel(R) Core(TM) i5-2320 CPU, 3 GHz, 6 GB of RAM. The platform used has been Ubuntu 12.04.

First of all, the parameters of the algorithm are adjusted. Secondly, five different experiments are executed to analyse the effect of adding to the standard VRPTW, the real-world constraints suggested by the company. Furthermore, a comparative analysis with the best results from the literature corresponding to the standard VRPTW Solomon benchmark instances[1] is performed. Finally, results corresponding to real instances are also analysed.

4.1. Parameter setting

This section reports the values selected for the parameters that appear in the GVNS implemented in this work. The parameter setting for the algorithms has been done using the Friedman test (Daniel, 1990), which provides the best configuration of parameters and detects differences or equalities among configurations. The test instances used to do this parameter setting have been chosen among the Solomon instances with 100 customers. Two instances of each category have been randomly selected: *C*105, *C*107, *R*104, *R*109, *RC*102, *RC*106, *C*202, *C*203, *R*208, *R*210, *RC*203 and *RC*204. This nonparametric statistical test has also been useful to know the best combination of operators to be used in the local search phase of the GVNS algorithm.

Solomon heuristic provides a different solution for each execution because we have used a parameter $\alpha = 2$ in order to select the seed for each route. The movement operators used by the GVNS algorithm also need some parameters, which have been set. These parameters are used by movement operators that involve two routes. Regarding the first parameter, when it is necessary to select the second route to make the movement, this is selected among the $\beta = 3$ closest routes to the previous one. The second parameter is used to select a customer to be deleted from a route. This customer is selected among the $\gamma = 3$ closest feasible customers to the route where it will be inserted. The third parameter is used to select a customer from a route after which a previous chosen one could be inserted. This customer is selected among the $\lambda = 3$ closest feasible customers. Moreover, the number of iterations, M, that the shaking, local search and move decision processes are carried out in each iteration of the GVNS has been fixed to $M = 20$. Finally, as mentioned above, the general algorithm consists of repeating the Solomon and GVNS algorithms for $N = 10$ iterations.

4.2. Constraint effects

Since the problem tackled in this paper takes into account several constraints and the instances in the literature are not prepared to consider all of them together, a set of experiments has been carried out using 16 test problem instances based on the real data provided by a company in the Canary Islands. These instances, which consist of 100 customers, have been used to show the different behaviours obtained by the algorithm depending on the constraints that are taken into consideration. They are available in https://sites.google.com/site/gciports/vrptw/hfvrptw.

The first experiment reported in this section corresponds to the selection of the principal objective function to be minimized: Total_Distance, Time_Balance or Distance_Balance. In order to carry out this experiment, 4 instances have been used, which do not include any additional constraint to the standard VRPTW. On one hand, if the objective function Total_Distance is chosen, the total travelled distance will be optimized, but the balanced time obtained will be worse than the one obtained using Time_Balance as objective function, and the balanced distance will be worse than the one obtained using Distance_Balance as objective function. On the other hand, if either the objective function Time_Balance or Distance_Balance is chosen, all vehicles will be used, such as required by the company. In case of Time_Balance, the balancing of time among vehicles will be optimized, and in case of Distance_Balance, the balancing of travelled distance will be optimized. For the next experiments, the Total_Distance objective function is used, since it is usually the most common and demanded one.

Table 1 summarizes the results obtained from the analysis of the principal objective function. The first two columns indicate the instance under consideration (N1_1–N1_4) and the principal objective function that is selected by the companies; Total_Distance, Time_Balance and Distance_Balance. Columns 3–9 report the number of postponed services (*PS*), extra time (*ET*) required by the vehicles, time balance (*TB*), distance balance (*DB*), cost (*C*), total distance (*D*) and number of routes (*NR*) in the obtained solutions. Time is expressed in seconds and distance in meters. As shown in Table 1, in general, when an objective function is selected as principal one, the best value of this objective is achieved in each case. Moreover, the best cost is always obtained using the Total_Distance objective function, since the total time spent doing less routes is usually lower.

[1] http://web.cba.neu.edu/~msolomon/problems.htm.

Table 1
Effects of the principal objective function.

Instance	Principal Objective	*PS*	ET	*TB*	*DB*	*C*	*D*	*NR*
N1_1	Total_Distance	0	0	33 602	231 363	**5834.08**	**1.0323e+06**	7
	Time_Balance	0	0	**6779**	203 177	7490.67	2.11398e+06	10
	Distance_Balance	0	0	23 485	**1261**	7407.17	2.16066e+06	10
N1_2	Total_Distance	0	0	17 837	223 827	**5054.83**	**862 046**	6
	Time_Balance	0	0	**484**	370 569	6855.08	1.94522e+06	10
	Distance_Balance	0	0	21 989	**7738**	7231.36	2.17563e+06	10
N1_3	Total_Distance	0	0	17 354	243 031	**5474.00**	**1.03364e+06**	6
	Time_Balance	0	0	**1256**	256 063	7406.53	1.69382e+06	10
	Distance_Balance	0	0	49 800	**2331**	7893.19	2.14595e+06	10
N1_4	Total_Distance	0	0	14 887	201 456	**5300.86**	**962 762**	7
	Time_Balance	0	0	**377**	223 874	6679.25	1.98046e+06	10
	Distance_Balance	0	0	26 022	**6770**	7242.97	1.89009e+06	10

PS: postponed services; *ET*: extra time; *TB*: time balance; *DB*: distance balance; *C*: cost; *D*: distance; *NR*: number of routes.

Table 2
Effects of using a heterogeneous fleet of vehicles with different capacities.

Instance	Vehicle capacity	*PS*	*ET*	*TB*	*C*	*D*	*NR*
N2_1	High	0	0	16 541	6099.47	**989 247**	7
	Low	0	0	45 605	8383.39	1.21453e+06	11
	Mixed 1	0	0	38 572	6322.69	1.0509e+06	7
	Mixed 2	0	0	44 478	7694.36	1.09895e+06	10
N2_2	High	0	0	15 062	5044.47	**858 015**	6
	Low	0	0	23 876	7829.89	1.34123e+06	12
	Mixed 1	0	0	21 769	5333.61	860 895	6
	Mixed 2	0	0	30 944	5745.78	934 271	9
N2_3	High	0	0	37 435	6146.00	**867 274**	6
	Low	0	0	19 955	8202.14	1.10491e+06	11
	Mixed 1	0	0	21 534	5924.33	879 834	7
	Mixed 2	0	0	26 135	6824.64	961 474	9
N2_4	High	0	0	18 032	5320.81	**96 9013**	7
	Low	0	0	20 459	8800.92	1.34612e+06	12
	Mixed 1	0	0	14 304	5668.25	1.00986e+06	7
	Mixed 2	0	0	21 417	6617.17	1.06245e+06	10

PS: postponed services; *ET*: extra time; *TB*: time balance; *C*: cost; *D*: distance; *NR*: number of routes.

The second experiment summarized in Table 2 corresponds to the use of a heterogeneous fleet of vehicles regarding capacity. This means that the capacities of the used vehicles are different. First and second columns indicate the instance and the vehicle capacity, respectively. Columns 3–8 report the number of postponed services (*PS*), extra time (*ET*) required by the vehicles, time balance (*TB*), cost (*C*), total distance (*D*) and number of routes (*NR*) in the obtained solutions. If we use vehicles with a high capacity (i.e. 500), we will need a small number of vehicles, but when the capacity is lower (i.e. 200), it is clear that we will need more vehicles, and therefore, more routes to serve all customers. However, when there is a mix of vehicles with different capacities (i.e. a half of 500 and a half of 200, which corresponds to the case of *Mixed* 1 in the table), the results will involve the vehicles with a higher capacity and there will be a smaller number of routes. If the number of vehicles with high capacity is reduced, i.e. three vehicles of 500 and the rest of vehicles of 200, which corresponds to the case of *Mixed* 2 in the table, the vehicles with higher capacity will be used first and then the rest of vehicles, and the number of routes will be intermediate. In any case, using vehicles with high capacity is associated with a shorter travelled distance.

The third experiment, which is reported in Table 3, corresponds to the use of a heterogeneous fleet with regard to the working shifts. Columns 1 and 2 show the instance and the characteristics of the working shift selected for the vehicles, respectively. Columns 3–8 report the number of postponed services (*PS*), extra time (*ET*) required by the vehicles, time balance (*TB*), cost (*C*), total distance (*D*) and number of routes (*NR*) in the obtained solutions. If the vehicles have restricted working shifts (narrow and multiple time intervals within working shifts) and extra hours are permitted (case *Restricted windows*, *extra time*), the results will contain extra hours and more routes than necessary in the case of homogeneous fleet. Nevertheless, if extra hours are not allowed, but the priorities do allow postponing customers for the next day (case *Restricted windows*, *postponing*), the routes will not include the postponed customers. If we compare these first two cases with restricted windows, we observe that postponing customers when it is possible involves lower travelled distance and cost. However, the number of postponed customers is very small in relation to the total number of customers, so that this last option could be appropriate for some companies which can afford this delay in service. Other tested case with regard to vehicles working shifts consists of having three vehicles with short working shifts, five with medium working shifts, and five with large working shifts (case *Mixed windows*). This way, the routes will be done by the vehicles with larger working shifts, involving the lowest number of routes and distance.

The fourth experiment reported in Table 4 corresponds to the use of narrow time windows for most of the customers. First and second columns indicate the instance and the features of the time windows selected for the customers, respectively. Columns 3–8 report the number of postponed services (*PS*), extra time (*ET*) required by the vehicles, time balance (*TB*), cost (*C*), total distance (*D*) and number of routes (*NR*). In case of using restricted customer windows (case *Restricted customer windows*) when priorities allow postponing customers, the number of postponed services increases, although the total travelled distance also increases with regard to the case without restricted customer windows (case *Non-restricted customer windows*) due to difficulties in compliance with customers time windows when trying to visit them in the same route.

4.3. Comparative with the literature

This section summarizes the comparison over the standard Solomon instances of 100 customers. Note that the algorithm proposed in this work is designed to solve the RVRPTW with the inclusion of all the real-world constraints explained in previous sections. Therefore, the method is not supposed to be the most competitive over these instances, particularly due to the fact that real instances have different features that have to be tested in computational time. Table 5

Table 3
Effects of using a heterogeneous fleet of vehicles with different working shifts.

Instance	Vehicle time window	*PS*	*ET*	*TB*	*C*	*D*	*NR*
N3_1	Restricted windows, extra time	0	2039	21 172	6278.89	1.11156e+06	10
	Restricted windows, postponing	3	0	14 549	5907.03	1.02771e+06	8
	Mixed windows	0	0	27 252	5777.33	**1.02485e+06**	**7**
N3_2	Restricted windows, extra time	0	1222	21 846	5837.17	1.45060e+06	10
	Restricted windows, postponing	2	0	25 974	5418.86	1.17176e+06	9
	Mixed windows	0	0	21 549	5382.89	**856 805**	**6**
N3_3	Restricted windows, extra time	0	2073	11 917	5246.50	1.01146e+06	8
	Restricted windows, postponing	2	0	26 434	5099.69	969 548	8
	Mixed windows	0	0	31 533	6008.39	**870 132**	**7**
N3_4	Restricted windows, extra time	0	1038	21 758	5998	1.29676e+06	10
	Restricted windows, postponing	1	0	28 301	5904.5	1.14511e+06	9
	Mixed windows	0	0	15 867	5263.69	**963 579**	**7**

PS: postponed services; *ET*: extra time; *TB*: time balance; *C*: cost; *D*: distance; *NR*: number of routes.

Table 4
Effects of using customers with restricted time windows.

Instance	Customer time windows	*PS*	*ET*	*TB*	*C*	*D*	*NR*
N4_1	Non-restricted customer windows	**0**	0	33 602	5834.08	**1.0323e+06**	7
	Restricted customer windows	5	0	33 382	9265.47	1.66452e+06	10
N4_2	Non-restricted customer windows	**0**	0	17 837	5054.83	**862 046**	6
	Restricted customer windows	3	0	35 661	9829.08	1.63362e+06	10
N4_3	Non-restricted customer windows	**0**	0	17 354	5474.00	**1.03364e+06**	6
	Restricted customer windows	5	0	26 273	10 676.2	1.54976e+06	10
N4_4	Non-restricted customer windows	**0**	0	14 887	5300.86	**962 762**	7
	Restricted customer windows	3	0	25 024	7086.28	1.52592e+06	10

PS: postponed services; *ET*: extra time; *TB*: time balance; *C*: cost; *D*: distance; *NR*: number of routes.

Table 5
Computational results of the standard static Solomon instances.

	BKS		Hong (2012)			Gong et al. (2012)			GVNS					
Instances	*NV*	*TD*	*NV*	*TD*	*t(s.)*	*NV*	*TD*	*t(s.)*	*NV*	*TD*	*t(s.)*	Gap_1	Gap_2	Gap_3
C1	10.00	828.48	10.00	833.10	48.89	10.00	835.91	50.94	10.00	838.45	595.66	1.20	0.64	0.30
R1	11.92	1210.34	12.25	1218.28	78.09	12.58	1232.34	95.63	14.08	1263.07	637.66	4.36	3.68	2.49
RC1	11.50	1384.16	12.13	1369.57	58.27	12.13	1385.47	84.57	13.63	1409.31	715.86	1.82	2.90	1.72
C2	3.00	589.86	3.00	590.31	25.26	3.00	593.41	39.52	3.13	632.90	1050.97	7.30	7.21	6.65
R2	2.73	951.03	3.27	964.11	102.07	3.00	1016.66	134.73	5.00	1019.38	774.94	7.19	5.73	0.27
RC2	3.25	1119.24	3.75	1131.18	74.82	3.38	1169.07	132.65	6.00	1151.08	810.97	2.84	1.76	−1.54
Avg.												4.12	3.65	1.65

summarizes the comparative regarding the best known solutions (*BKS*) until 2009,[2] the results from Hong (2012), and the results from Gong et al. (2012). These two later works have been chosen as references because they provide better results than the best known until 2009 for some particular instances. The first column of the table shows the Solomon instance category. Then, the average values of number of vehicles (*NV*), traveled distance (*TD*) and computational time (*t(s.)*) are reported for each approach. Computational times have been scaled using the computer speed measured in millions of floating-point operations per second (Mflop/s) in order to make a fair comparison. Finally, Gap_1, Gap_2 and Gap_3 correspond to the gap (%) between *BKS* and our proposal, between Hong (2012) results and our proposal, and between Gong et al. (2012) results and our proposal, respectively. As can be seen, the average gap regarding *BKS* is about 4% and lower regarding the other later approaches.

Notice that this paper is not aimed at overcoming the best results for the standard Vehicle Routing Problem with Time Windows, but rather at proposing an intelligent algorithm that works effectively when real-world instances with real-world constraints are solved. However, the GVNS algorithm provides solutions that overcome the best known literature results for some particular instances, and, in average, its solutions are very close to these best known. A total average of 12 min are needed to obtain the result of a Solomon instance, with an average deviation of 3 min.

Although this has been the solution proposed to the real company, other approaches can be developed in order to improve the average deviation for the standard instances which do not include all the real-world constraints. Until the moment, the used VND process has made movements in or between routes choosing routes and customers randomly. However, it is possible to explore

[2] http://www.sintef.no/Projectweb/TOP/VRPTW/Solomon-benchmark/100-customers/.

the search space more deeply in order to choose the movement of customers which involves the shortest distance. This process would take much more time, and for this reason it would not be suitable for the real company. Nevertheless, a combination of both processes can be carried out trying to keep similar times. Furthermore, we can stop the search with the first best solution found.

Taking into account all these aspects, three more approaches have been developed. Parameters M and N have been adjusted using Friedman test, among the values that keep similar times to the original process. For the first tested approach, the relocation movement has been applied in its best first solution version, instead of the original VND. Parameters N and M have been set to 5 and 7, respectively. Results are shown in Table 6. The average gap regarding the best known solutions decreases from 4.12 to 1.70 using this approach, and even results from Gong et al. (2012) are improved, demonstrating that it is possible to get better behaviour with the standard instances in the literature which do not include all the real-world constraints.

The second tested approach applies the relocation movement in its greedy version and the original VND half and half. Parameters N and M have been set to 2 and 9 respectively. This way, results in Table 7 are obtained, where the average gap regarding the best known solutions is reduced to 0.64 for the standard instances. These results are better than the ones obtained if the relocation movement, in its greedy version, is applied only at some steps of the GVNS, for example, when $k=3$ or $k=6$ in the shaking process. In this case, parameters N and M have been set to 3 and 9 respectively. Table 8 depicts these results, where the average gap regarding the best known solutions is 2.69.

Therefore, the best approach to use with the standard instances in the literature is the one that combines the original VND proposed in this paper and the greedy version of the relocation movement, half and half.

Finally, it is important to note that the number of vehicles used is not important for the real company referenced in this paper, since it has a fixed fleet of vehicles with a fixed number of drivers. However, if the number of vehicles had to be reduced, a number of routes reduction algorithm would have to be applied before the GVNS algorithm. This process usually involves a high increase of computational time, and for this reason it is not suitable for the real company, whose main requirement is to obtain a fast solution with a high quality. Nevertheless, the reduction of routes process proposed by Nagata and Brysy (2009) has been implemented in order to check this option. For this purpose, the parameter setting has been made taking into account that both together, reduction of routes algorithm and the original algorithm proposed in this paper, do not overpass the times of the original approach. Results for the standard static Solomon instances are shown in Table 9. Distance

Table 6
Computational results of the standard static Solomon instances applying first best solution approach.

	BKS		Hong (2012)		Gong et al. (2012)		GVNS				
Instances	*NV*	*TD*	*NV*	*TD*	*NV*	*TD*	*NV*	*TD*	Gap_1	Gap_2	Gap_3
C1	10.00	828.48	10.00	833.10	10.00	835.91	10.00	837.28	1.06	0.50	0.16
R1	11.92	1210.34	12.25	1218.28	12.58	1232.34	13.91	1233.55	1.92	1.25	0.10
RC1	11.50	1384.16	12.13	1369.57	12.13	1385.47	13.50	1395.71	0.83	1.91	0.74
C2	3.00	589.86	3.00	590.31	3.00	593.41	3.25	619.32	4.99	4.91	4.37
R2	2.73	951.03	3.27	964.11	3.00	1016.66	5.90	975.47	2.57	1.18	−4.05
RC2	3.25	1119.24	3.75	1131.18	3.38	1169.07	6.25	1105.87	−1.19	−2.24	−5.41
Avg.									1.70	1.25	−0.68

Table 7
Computational results of the standard static Solomon instances applying best solution and original VND half and half.

	BKS		Hong (2012)		Gong et al. (2012)		GVNS				
Instances	*NV*	*TD*	*NV*	*TD*	*NV*	*TD*	*NV*	*TD*	Gap_1	Gap_2	Gap_3
C1	10.00	828.48	10.00	833.10	10.00	835.91	10.00	833.62	0.62	0.06	−0.27
R1	11.92	1210.34	12.25	1218.28	12.58	1232.34	13.83	1230.08	1.63	0.97	−0.18
RC1	11.50	1384.16	12.13	1369.57	12.13	1385.47	13.62	1395.82	0.84	1.92	0.75
C2	3.00	589.86	3.00	590.31	3.00	593.41	3.12	611.32	3.64	3.56	3.02
R2	2.73	951.03	3.27	964.11	3.00	1016.66	5.27	956.27	0.55	−0.81	−5.94
RC2	3.25	1119.24	3.75	1131.18	3.38	1169.07	6.12	1080.76	−3.44	−4.46	−7.55
Avg.									0.64	0.21	−1.70

Table 8
Computational results of the standard static Solomon instances applying best solution for $k=3$ and $k=6$.

	BKS		Hong (2012)		Gong et al. (2012)		GVNS				
Instances	*NV*	*TD*	*NV*	*TD*	*NV*	*TD*	*NV*	*TD*	Gap_1	Gap_2	Gap_3
C1	10.00	828.48	10.00	833.10	10.00	835.91	10.00	849.92	2.59	2.02	1.68
R1	11.92	1210.34	12.25	1218.28	12.58	1232.34	14.08	1236.74	2.18	1.52	0.36
RC1	11.50	1384.16	12.13	1369.57	12.13	1385.47	13.63	1404.58	1.47	2.56	1.38
C2	3.00	589.86	3.00	590.31	3.00	593.41	3.13	629.78	6.77	6.69	6.13
R2	2.73	951.03	3.27	964.11	3.00	1016.66	5.64	987.20	3.80	2.40	−2.90
RC2	3.25	1119.24	3.75	1131.18	3.38	1169.07	6.13	1111.60	−0.68	−1.73	−4.92
Avg.									2.69	2.24	0.29

Table 9
Computational results of the standard static Solomon instances applying routes reduction algorithm.

	BKS		Hong (2012)		Gong et al. (2012)		GVNS				
Instances	*NV*	*TD*	*NV*	*TD*	*NV*	*TD*	*NV*	*TD*	Gap_1	Gap_2	Gap_3
C1	10.00	828.48	10.00	833.10	10.00	835.91	10.00	841.37	1.56	0.99	0.65
R1	11.92	1210.34	12.25	1218.28	12.58	1232.34	13.08	1224.68	1.19	0.53	−0.62
RC1	11.50	1384.16	12.13	1369.57	12.13	1385.47	12.87	1405.49	1.54	2.62	1.45
C2	3.00	589.86	3.00	590.31	3.00	593.41	3.12	621.50	5.36	5.28	4.73
R2	2.73	951.03	3.27	964.11	3.00	1016.66	3.36	1079.87	13.55	12.01	6.22
RC2	3.25	1119.24	3.75	1131.18	3.38	1169.07	4.12	1245.58	11.29	10.11	6.54
Avg.									5.75	5.26	3.16

Table 10
Best solutions reached for the real instance using the total distance as principal objective function.

Solution	Distance	Time balance	Distance balance	Nr. routes
GVNS sol.	852.7	37 200	196.5	7
GVNS sol.	858.9	12 840	125.3	6
Real sol.	1247.3	14 040	228.5	6

Table 11
Comparison between initial solution and new constraints solution.

Solution	Distance	Time balance	Distance balance	Nr. routes
GVNS initial sol.	852.7	37 200	196.5	7
GVNS new constraints sol.	1154.5	19 800	199.7	13

deviations are not good enough due to the short computational time which can be used. The time needed for both the reduction of routes algorithm and the original GVNS algorithm to obtain high quality solutions need to be increased, but, as said before, the real company requires a quick algorithm and has a fixed fleet of vehicles. Therefore, this reduction of routes has not been included in the final algorithm.

4.4. Results for real instances

This section is devoted to show results corresponding to real data. In the first place, the solutions provided by the proposed GVNS are compared with the solution implemented by a company for a real instance, which consists of 109 customers. Table 10 summarizes the best solutions obtained by GVNS when considering the total distance objective function, and the best solution implemented by the company. Moreover, the time/distance balance associated to the solutions are also reported. The first row of the table corresponds to the best solution obtained by GVNS if the total distance is considered as objective function. The second row shows the solution obtained by GVNS consisting of 6 routes, which is the number of vehicles of the company solution. The third row shows the company solution.

From these results, we may conclude that the solution implemented by the company is worst than the solutions obtained by GVNS if the total distance objective and the number of routes are taken into consideration. Note that our second obtained solution is better in total distance, time and distance balance than the one that had been implemented by the company, and our first obtained solution provides the lowest total distance although it presents the worst time balance.

Additionally, the company plans to change its fleet of vehicles, so that it pretends to have eight vehicles with capacity 100, four vehicles with capacity 200 and four vehicles with capacity 300. First kind of vehicles will be available all day, second kind will be available in two working shifts and the third kind of vehicles will only be available in the morning. This last kind of vehicle will not be able to serve some customers due to road size restrictions. With this plan in mind, the company wants to know how this would affect the vehicles' routes. Taking all these aspects into account, the new instance has been solved using the GVNS and results indicate that the distance and number of routes will increase regarding the initial solution with initial fleet, as shown Table 11. Therefore, getting this algorithm is also useful to predict future situations and help decision maker.

Finally, since the image corresponding to the solution $N1$ is confusing due to the large number of routes, the solution to an instance based on $N1$ consisting of 50 customers is shown in Fig. 1, in which its 5 routes are depicted over the map of the island of Tenerife.

5. Conclusions

This work tackles a real-world Rich Vehicle Routing Problem (*RVRPTW*) proposed by a company in Spain that combines multiple attributes together, which aim to better consider the specificities of real applications. Particularly, it has developed an efficient solution method to solve the *RVRPTW*. The attributes considered by the company consist of a fixed heterogeneous fleet of vehicles, soft and multiple time windows for customers, soft and multiple time intervals in the working shifts for vehicles, customers' priorities and vehicle–customer constraints, which to the best of our knowledge have not been taken into account all together in the literature. Given the fact that the company might require to either consider or not all these attributes, the fleet management system allows deactivating any of the mentioned attributes. Since exact algorithms are not applicable when solving real-life large vehicle routing problem instances, the focus of this work is put on the use of metaheuristic procedures. Particularly, this paper proposes a General Variable Neighbourhood Search (*GVNS*) metaheuristic, which is able to obtain high quality solutions that are valid for the company. The computational experience carried out in this work, which includes the analysis of the effect of the different attributes taken into consideration, the comparative with the literature for the standard vehicle routing problem with time windows, and the study of the solutions provided by the algorithm when compared with the solutions implemented by the company, corroborate the effectiveness of the developed software. It is worth to mention that the algorithm has been integrated into a fleet management system and several tests with real companies have been conducted. Finally, the next phase of this work is to dynamically route new customers which appear over the planning horizon. It means that, once the initial routes have been specified

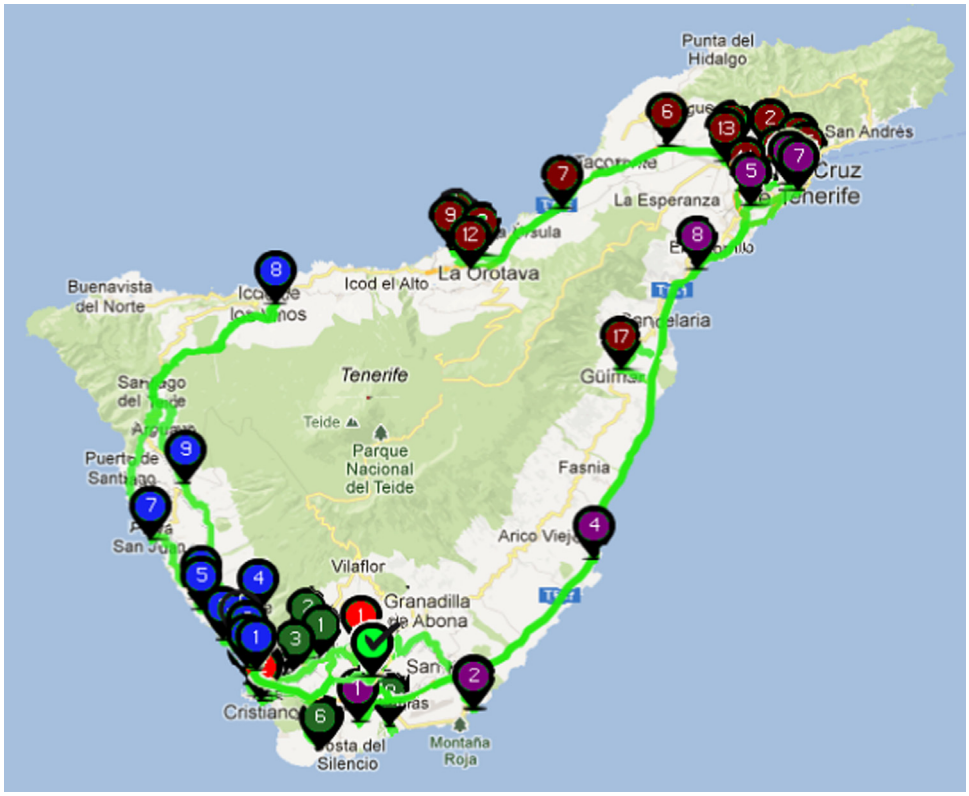

Fig. 1. Some routes corresponding to a real instance.

and vehicles start to make their work, new customers can ask for service. At this point real time communication between vehicles and the fleet management system is required.

Acknowledgements

This work has been partially funded by the Spanish Ministry of Economy and Competitiveness (Project TIN2012-32608) and the Spanish Ministry of Industry, Tourism and Trade (Project TSI-020100-2011-298).

References

Baldacci, R., Battarra, M., Vigo, D., 2008. Routing a heterogeneous fleet of vehicles. In: The Vehicle Routing Problem: Latest Advances and New Challenges, Springer, US, pp. 3–27.

Baldacci, R., Mingozzi, A., 2009. A unified exact method for solving different classes of vehicle routing problems. Math. Program. 120 (2), 347–380.

Baldacci, R., Toth, P., Vigo, D., 2007. Recent advances in vehicle routing exact algorithms. 4OR 5 (4), 269–298.

Borgulya, I., 2008. An algorithm for the capacitated vehicle routing problem with route balancing. Centr. Eur. J. Oper. Res. 16 (4), 331–344.

Brandão, J., 2011. A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. Comput. Oper. Res. 38 (1), 140–151.

Bräysy, O., 2003. A reactive variable neighborhood search for the vehicle routing problem with time windows. INFORMS J. Comput. 15, 347–368.

Bräysy, O., Gendreau, M., 2005. Vehicle routing problem with time windows, Part ii: metaheuristics. Transp. Sci. 39 (1), 119–139.

Calvete, H.I., Gale, C., Oliveros, M.-J., Sanchez-Valverde, B., 2007. A goal programming approach to vehicle routing problems with soft time windows. Eur. J. Oper. Res. 177 (3), 1720–1733 (6th International Conference on Multiple Objective Programming and Goal Programming, Hammamet, Tunisia, April 04–06, 2004).

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

Croes, G., 1958. A method for solving traveling-salesman problems. Oper. Res. 6 (6), 791–812.

Daniel, W.W., 1990. Applied Nonparametric Statistics. PWS-Kent Publishing Company, Boston.

Eksioglu, B., Vural, A.V., Reisman, A., 2009. Survey: the vehicle routing problem: a taxonomic review. Comput. Ind. Eng. 57 (4), 1472–1483.

Fleszar, K., Osman, I., Hindi, K., 2008. A variable neighbourhood search algorithm for the open vehicle routing problem. Eur. J. Oper. Res. 195 (3), 803–809.

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

Gendreau, M., Potvin, J.-Y., Bräumlaysy, O., Hasle, G., Løkketangen, A., 2008. Metaheuristics for the Vehicle Routing Problem and its Extensions: A Categorized Bibliography. Springer, US.

Gendreau, M., Tarantilis, C.D., 2010. Solving large-scale vehicle routing problems with time windows: the state-of-the-art. CIRRELT.

Ghoseiri, K., Ghannadpour, S.F., 2010. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. Appl. Soft Comput. 10 (4), 1096–1107.

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. 13 (5), 533–549.

Goel, A., Gruhn, V., 2008. A general vehicle routing problem. Eur. J. Oper. Res. 191 (3), 650–660.

Golden, B.L., Raghavan, S., Wasil, E.A., 2008. The Vehicle Routing Problem: Latest Advances and New Challenges. vol. 43. Springer, US.

Gong, Y.-J., Zhang, J., Liu, O., Huang, R.-Z., Chung, H.-H., Shi, Y., 2012. Optimizing the vehicle routing problem with time windows: a discrete particle swarm optimization approach. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. 42 (2), 254–267.

Hansen, P., Mladenovic, N., Brimberg, J., Moreno-Pérez, J., 2010a. Variable neighborhood search. In: Handbook of Metaheuristics. Springer, US, pp. 61–86.

Hansen, P., Mladenovic, N., Moreno-Pérez, J., 2008. Variable neighborhood search. Eur. J. Oper. Res. 191 (3), 593–595.

Hansen, P., Mladenovic, N., Moreno-Pérez, J.A., 2010b. Variable neighbourhood search: methods and applications. Ann. Oper. Res. 175 (1), 367–407.

Hemmelmayr, V., Doerner, K., Hartl, R., 2009. A variable neighborhood search heuristic for periodic routing problems. Eur. J. Oper. Res. 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. Eur. J. Oper. Res. 191 (3), 691–704.

Hong, L., 2012. An improved LNS algorithm for real-time vehicle routing problem with time windows. Comput. Oper. Res. 39 (2), 151–163.

Hong, S., Park, Y., 1999. A heuristic for bi-objective vehicle routing with time window constraints. Int. J. Prod. Econ. 62 (3), 249–258.

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. Transp. Sci. 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 Appl. Math. 156 (11), 2050–2069.

Jozefowiez, N., Semet, F., Talbi, E.G., 2007. From single-objective to multi-objective vehicle routing problems: motivations, case studies and methods. In: Golden, B., Raghavan, S., Wasil, E. (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, US.

Jozefowiez, N., Semet, F., Talbi, E.-G., 2008. Multi-objective vehicle routing problems. Eur. J. Oper. Res. 189 (2), 293–309.

Kritikos, M., Ioannou, G., 2010. The balanced cargo vehicle routing problem with time windows. Int. J. Prod. Econ. 123 (1), 42–51.

Kytöjoki, J., Nuortio, T., Brysy, O., Gendreau, M., 2007. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. Comput. Oper. Res. 34 (9), 2743–2757.

Lahyani, R., Khemakhem, M., Semet, F., 2015. Rich vehicle routing problems: from a taxonomy to a definition. Eur. J. Oper. Res. 241 (1), 1–14.

Laporte, G., 2009. Fifty years of vehicle routing. Transp. Sci. 43 (4), 408–416.

Li, F., Golden, B., Wasil, E., 2007. A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. Comput. Oper. Res. 34 (9), 2734–2742.

Melian, B., 2006. Using memory to improve the VNS metaheuristic for the design of SDH/WDM networks. In: Almeida, F, Aguilera, M.J.B., Blum, C., Vega, J.M.M., Perez, M.P., Roli, A., Sampels, M. (Eds.), Hybrid Metaheuristics, Proceedings. Lecture Notes in Computer Science. 3rd International Workshop on Hybrid Metaheuristics, Gran Canaria, Spain, October 13–14, vol. 4030, pp. 82-93.

Melián-Batista, B., DeSantiago, A., Angelbello, F., Alvarez, A., 2014. A bi-objective vehicle routing problem with time windows: a real case in Tenerife. Appl. Soft Comput. J. 17, 140–152.

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

Nagata, Y., Bräysy, O., 2009. A powerful route minimization heuristic for the vehicle routing problem with time windows. Oper. Res. Lett. 37 (5), 333–338.

Or, I., 1976. Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking. Xerox University Microfilms.

Paraskevopoulos, D., Repoussis, P., Tarantilis, C., Ioannou, G., Prastacos, G., 2008. A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows. J. Heuristics 14 (5), 425–455.

Pardo, E., Mladenovi, N., Pantrigo, J., Duarte, A., 2013. Variable formulation search for the cutwidth minimization problem. Appl. Soft Comput. J. 13 (5), 2242–2252.

Penna, P.H.V., Subramanian, A., Ochi, L.S., 2011. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. J. Heuristics, 1–32.

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

Potvin, J.-Y., 2009. State-of-the art review—evolutionary algorithms for vehicle routing. INFORMS J. Comput. 21 (4), 518–548.

Prins, C., 2009. Two memetic algorithms for heterogeneous fleet vehicle routing problems. Eng. Appl. Artif. Intell. 22 (6), 916–928.

Rahoual, M., Kitoun, B., Mabed, M., Bachelet, V., Benameur, F., 2001. Multicriteria genetic algorithms for the vehicle routing problem with time windows. In: Metaheuristic International Conference (MIC'2001), pp. 527–532.

Repoussis, P.P., Paraskevopoulos, D.C., Tarantilis, C.D., Ioannou, G., 2006. A reactive greedy randomized variable neighborhood tabu search for the vehicle routing problem with time windows. In: Almeida, F., Aguilera, M.J.B., Blum, C., Vega, J.M. M., Perez, M.P., Roli, A., Sampels, M. (Eds.), Hybrid Metaheuristics, Proceedings. Lecture Notes in Computer Science, 3rd International Workshop on Hybrid Metaheuristics, Gran Canaria, Spain, October 13–14, vol. 4030, pp. 124–138.

Schmid, V., Doerner, K.F., Laporte, G., 2013. Rich routing problems arising in supply chain management. Eur. J. Oper. Res. 224 (3), 435–448.

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

Subramanian, A., Vaz Penna, P.H., Uchoa, E., Ochi, L.S., 2012. A hybrid algorithm for the heterogeneous fleet vehicle routing problem. Eur. J. Oper. Res. 221 (2), 285–295.

Taillard, E., 1999. A heuristic column generation method for the heterogeneous fleet vrp. Rairo-Rech. Oper.-Oper. Res. 33 (1), 1–14.

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. Transp. Sci. 31 (2), 170–186.

Tarantilis, C.D., Zachariadis, E.E., Kiranoudis, C.T., 2009. A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem. IEEE Trans. Intell. Transp. Syst. 10 (2), 255–271.

Toth, P., Vigo, D., 2002. The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, US.

Vidal, T., Crainic, T., Gendreau, M., Prins, C., 2013. Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. Eur. J. Oper. Res. 231 (1), 1–21.