Journal of the Operational Research Society (2012) 63, 569–581

www.palgrave-journals.com/jors/

Optimizing vehicle routes in a bakery company allowing flexibility in delivery dates

J Pacheco[1*], A Alvarez[2], I García[2,3] and F Angel-Bello[4]

[1]*Universidad de Burgos, Burgos, España;* [2]*Ciudad Universitaria, Monterrey, Nuevo León, México;* [3]*Universidad Autónoma de Coahuila, Saltillo, México; and* [4]*Instituto Tecnológico de Monterrey, Monterrey, México*

The work addressed in this paper is motivated from a real problem proposed to the authors by a bakery company in Northern Spain. The objective is to minimize the total distance travelled for the daily routes over the week. In order to reduce this total distance, some flexibility in the dates of delivery is introduced. A mixed-integer linear model for the problem is formulated. In addition, a two-phase method based in GRASP and path-relinking metaheuristic strategies is proposed. Computational experiments show that the method performs very well, obtaining high-quality solutions in short computational times. Moreover, when it is applied to real-data-based instances, the obtained solutions considerably reduce transportation costs over the planning horizon.

Journal of the Operational Research Society (2012) **63,** 569–581. doi:10.1057/jors.2011.51
Published online 29 June 2011

Keywords: VRP with flexibility in delivery; metaheuristics; GRASP; path relinking

1. Introduction

The vehicle routing problem, VRP, is undoubtedly one of the most studied combinatorial optimization problems in the literature. It can be defined as the problem of designing routes for delivering vehicles of given capacities, to supply a set of customers with known locations and demands from a single depot. Routes for the vehicles are designed to minimize some objective such as the total distance travelled.

In general, each client must be visited once; routes must begin and end in a central depot and capacity restrictions of the vehicle must be respected. Besides the basic VRP, many variants may appear since there are many possibilities in real-life problem settings, for example: pickup and delivery, backhauling, multiple depots, heterogeneous fleet, multiple routes per vehicle, etc. There are many works devoted to the exact and approximate solution for the several variants of this problem. For general surveys of VRP, we refer to Toth and Vigo (2002) and Bräysy and Gendreau (2005); Cordeau *et al* (2007) provide a survey of methods for VRP with time windows.

As it is known, VRP is an NP-hard problem. Even though there are some techniques for solving it accurately (Fisher, 1994; Cordeau *et al*, 2007), the literature on heuristic techniques is much more extensive: from the classical algorithm of savings of Clarke and Wright (1964), the swap algorithm of Gillet and Miller (1974), or the interesting algorithm of Fisher and Jaikumar (1981) until the development of more modern metaheuristics. VRPs exhibit an impressive record of successful metaheuristic implementations, some of them are the following: genetic algorithms (Potvin and Bengio, 1994; Thangiah *et al*, 1994; Prins, 2004; Marinakis *et al*, 2007); temple simulated (Osman, 1993), tabu search (Gendreau *et al*, 1994; Rochat and Taillard, 1995; Taillard *et al*, 1997; Cordeau *et al*, 2001; Prins and Prodhon, 2007); GRASP (Kontoravdis and Bard, 1995); guided local search (Vondouris and Tsang, 1999); ant colony (Gambardella *et al*, 1999) or variable neighbourhood search (Bräysy, 2003; Hemmelmayr *et al*, 2009; Imram *et al*, 2009).

Other recent and interesting contributions for several VRP variants can be found in Dullaert *et al* (2002), Mester and Bräysy (2005), Haghani and Jung (2005), Ropke and Pisinger (2006), Tan *et al* (2006), Pisinger and Ropke (2007), Li *et al* (2007), Tarantilis and Kiranoudis (2007), Goel and Gruhn (2008), Jozefowiez *et al* (2008), Bolduc *et al* (2008), Flisberg *et al* (2009), Zachariadis *et al* (2009a, b), Gajpal and Abad (2010), Potvin and Naud (2011), etc.

The work addressed in this paper is motivated by a real problem proposed to the authors by a bakery company in Northern Spain. The company has to meet required orders, known in advance, for a set of distribution centres over a week using a fleet of homogeneous vehicles. Each order

*Correspondence: J Pacheco, Facultad de Ciencias Económicas y Empresariales, Universidad de Burgos, Plaza Infanta Elena s/n, 09001 Burgos, Spain.
E-mail: jpacheco@ubu.es

includes a number of required pallets and a specific delivery day (*deadline*). Each trip finishes on the same day it started and the vehicles return to the depot at the end of the route. The way the problem is currently tackled by the company is by solving each day a classical VRP with the orders required for that day. Nevertheless, the company perceives that the costs could be further reduced.

The total travel costs over the week might be reduced by amalgamating orders, since it is possible to take advantage of certain flexibility in the delivery date. Nevertheless, as the company is dealing with perishable products, every order should not be delivered more than 1 day earlier than the original deadline. This policy will result in stock at the distributors, but it is controlled and acceptable for the company. Although initially this problem has been proposed by a bakery company it is clear that the same problem or similar can be found in other areas.

In this work we propose a mixed-integer linear formulation for the problem. In addition, a two-phase algorithm is designed to solve it, which is based on two metaheuristic strategies: GRASP and path relinking. Computational experiments show that the method performs very well, obtaining high-quality solutions in short computational times. Moreover, when it is applied to real-data-based instances, the obtained solutions considerably reduce transportation costs over the planning horizon.

The remainder of the paper is organized as follows. In Section 2, we describe the mathematical formulation proposed for this problem. In Section 3, the two-phase algorithm is explained in detail. Section 4 presents computational results using randomly generated and real-data-based instances. In addition, a set of real instances provided by the firm are analysed and solved. Finally, Section 5 concludes with our main findings and future research avenues.

2. Problem description and mathematical formulation

In this paper, we address a problem that arises in a bakery company, which has to deliver orders placed by geographically dispersed distribution centres in northern Spain. Orders for the week are known in advance, commonly on Friday of the previous week. The problem has the following characteristics.

Regarding orders:

- Each order includes a number of required pallets, which need to be delivery by a specific day (*deadline*).
- Each order must be served by a single vehicle (due to labour constraints at distribution centres).
- Each order should be delivered within the original deadline and no more than a fixed number of days before the deadline, but always within the same week. This means that if the deadline of an order is Monday, it cannot be delivered before.

Regarding routes:

- To meet the orders a fleet of homogeneous vehicles with known capacity is available.
- Each trip finishes on the same day it started and the vehicles return to the depot at the end of the route.
- Transportation costs are proportional to the distance travelled.

The problem consists of designing routes to deliver the orders placed for the week with the objective of minimizing the total transportation cost. Flexibility in the day of delivery plays an important role in the solution to the problem. Managing this flexibility, it is intended to create savings in transportation costs, and is achieved as we shall see below. Note that without it, that is, when each order must be delivered within its deadline, the problem would consist of solving five CVRP. Each CVRP is related to orders requested for each day of the week (from Monday to Friday). This is the way the company was addressing the problem.

To formalize the statement of the problem let us introduce the following notation.

H	is the number of days of the planning horizon;
n	is the number of orders;
V	$\{0, 1, \ldots, n\}$ set of locations; where 0 corresponds to the depot and i corresponds to the location of ith order, $\forall\ i=1, \ldots, n$ (we prefer to identify every location by its corresponding order to simplify further descriptions and notations);
d_{ij}	is distance between location i and j, $\forall i, j \in V$;
q_i	is the number of pallets of order i, $\forall i=1, \ldots, n$;
e_i	is the *deadline* of order i, $\forall i=1, \ldots, n$; $e_i \in \{1, \ldots, H\}$;
Q	is the capacity of vehicles (in pallets);
K	is size of the fleet, that is, number of available vehicles every day;
g	is the maximum number of days that an order could be served before its *deadline*.

As we have mentioned before, in real instances, the parameter H is taken equal to 5, where 1 corresponds to Monday and 5 to Friday.

With the aim of formulating, the problem as a mixed-integer linear model let us introduce the following decision variables:

x_{ij}	1, if location j is visited immediately after i in the same route; 0 otherwise; $\forall i, j \in V$;
y_{it}	1, if location i is visited in day t; 0 otherwise; $\forall i \in V \backslash \{0\}$; $\forall t=1, \ldots, H$;
z_{it}	1, if location i is visited in day t and is the first point visited in its route (ie right after the depot); 0 otherwise; $\forall i \in V \backslash \{0\}$; $\forall t=1, \ldots, H$.

To facilitate the subsequent formulation we have defined parameters m_{it} that indicate the days that each order can be served. More precisely, $m_{it}=1$ if order i can be served in day t (ie, if $\max\{1,e_i-g\}\leqslant t\leqslant e_i$); otherwise $m_{it}=0$. For example, let $H=5$, $g=1$ and suppose that the deadline of the order 1 is Wednesday, that is, $e_1=3$, then this order could be served on Tuesday or Wednesday, which means that $m_{11}=0$, $m_{12}=1$, $m_{13}=1$, $m_{14}=0$, and $m_{15}=0$. In the same way, if the deadline of order 2 is Monday, then $m_{21}=1$, $m_{22}=0$, $m_{23}=0$, $m_{24}=0$ y $m_{25}=0$.

The objective is

$$\min \sum_{i\in V}\sum_{j\in V} d_{ij}x_{ij}$$

subject to

$$\sum_{i\in V} x_{ij}=1 \quad \forall\, j\in V\backslash\{0\} \tag{1}$$

$$\sum_{j\in V} x_{ij}=1 \quad \forall\, i\in V\backslash\{0\} \tag{2}$$

$$\sum_{i\in V} x_{i0}=\sum_{j\in V} x_{0j} \tag{3}$$

$$u_i-u_j+Q\,x_{ij}\leqslant Q-q_j \quad \forall\, i,j\in V\backslash\{0\},\ i\neq j \text{ such that } q_i+q_j\leqslant Q \tag{4}$$

$$q_i\leqslant u_i\leqslant Q \quad \forall\, i\in \mathrm{V}\backslash\{0\} \tag{5}$$

$$y_{it}\leqslant m_{it} \quad \forall\, i\in \mathrm{V}\backslash\{0\}, \quad \forall\, t\in\{1,\ldots,H\} \tag{6}$$

$$\sum_{t=1}^{H} y_{it}=1 \quad \forall\, i\in V\backslash\{0\} \tag{7}$$

$$y_{it}-y_{jt}\leqslant 1-x_{ij} \quad \forall\, i,j\in V\backslash\{0\},\ i\neq j \tag{8}$$

$$z_{it}\geqslant y_{it}+x_{0i}-1 \quad \forall\, i\in V\backslash\{0\}, \quad \forall\, t\in\{1,\ldots,H\} \tag{9}$$

$$\sum_{i=1}^{n} z_{it}\leqslant K \quad \forall\, t\in\{1,\ldots,H\} \tag{10}$$

$$x_{ij}\in\{0,1\} \quad \forall\, i,j\in V \tag{11}$$

$$y_{it}\in\{0,1\} \quad \forall\, i\in V\backslash\{0\}; \quad \forall t=1,\ldots,H \tag{12}$$

$$z_{it}\geqslant 0 \quad \forall\, i\in V\backslash\{0\}; \quad \forall t=1,\ldots,H \tag{13}$$

Constraints (1) and (2) are the classical indegree and outdegree constraints. Analogously, equation (3) ensures that all vehicles leaving the depot return to the depot; Equations (4) and (5) are the alternative family of constraints equivalent to the capacity-cut constraints, where u_i, $i\in V\backslash\{0\}$, are auxiliary continuous variables representing the load of the vehicle after visiting customer i (Toth and Vigo, 2002). Constraints (6) and (7) establish that every order must be served in exactly 1 day belonging to the set of its feasible days; Equation (8) guarantees that orders in the same route have to be served in the same day; Equations (9) and (10) impose the maximum number of routes per day (size of fleet).

This model has been useful in the computational experiments to assess the performance of the solution method that we propose.

Note that constraints (1)–(5) are often used in CVRP formulations (Toth and Vigo, 2002). Constraints (6)–(10) have been added to deal with the specific problem we are addressing. From this perspective, it can be said that this model constitutes a generalization of CVRP. On the other hand, the problem could be modelled as well as a particular case of PVRP (Christofides and Beasley, 1984; Francis *et al*, 2008). To do that, it should be considered that every order has two possible schedules (each one defined by 1 day) and requires only one visit in the planning horizon. However, preliminary experiments showed that addressing the problem as a PVRP leads to a much more complex model.

3. Solution approach

Our solution procedure is a two-phase algorithm. In the first phase a set of good and diverse solutions are generated, based on GRASP metaheuristic. In the second phase, in order to improve them, path relinking is applied to each pair of solutions belonging to the set obtained in Phase I.

In the proposed algorithm all orders and all routes over the planning horizon are considered jointly, that is, not day by day. Thus, our algorithm treats the problem as a CVRP with two additional constraints, as it was shown in the previous section: (1) the orders belonging to the same route should have at least a common feasible date and (2) the number of routes assigned to each day must not exceed the fleet size. Initially, every route is assigned to the minimum *deadline* of the orders supplied by this route. Obviously, this constraint replaces the constraint regarding to the maximum number of routes over the planning horizon.

GRASP, or greedy randomized adaptive search procedure, is a metaheuristic strategy that constructs solutions by a controlled randomization and a greedy function. Most GRASP implementations also include a local search that is used to improve upon the solutions generated with the randomized greedy function. GRASP was originally proposed in the context of a set covering problem (Feo and Resende, 1989). Details of the methodology and a survey of applications can be found in Feo and Resende (1995) and Pitsoulis and Resende (2002).

An outline of the first phase of the proposed algorithm is shown below.

Phase I (GRASP)

Repeat

1. Build a solution using a **Random Greedy Constructive Method**

2. Improve this solution by a **Local Search Method**

until a stop criterion is satisfied

In our implementation, the stop criterion is reached after a pre-fixed number of iterations.

During executing this phase, a set *Set_of_Best* with the best generated solutions is created. Let us denote as *n_best* the size of this set.

Path relinking is a strategy traditionally associated with the intensification phase in tabu search. The underlying idea is that in the *path* between two good solutions there should be solutions of similar quality (in some cases, even better solutions). See Glover *et al* (2000) for more details.

The second phase of the proposed algorithm is based on path-relinking strategy and can be shown as follows.

Phase II (Path Relinking)

For each pair of solutions S, S'∈ Set_of_Best do:
— *Generate a set of solutions in the path between S and S' using a* **Path Generator Method**
— *Improve these solutions by a* **Local Search Method**

In the following subsections, a detailed description of the procedures involved in these two phases is given, that is, the random greedy constructive method, local search method, and path generator method.

3.1. Random greedy constructive method

The designed *random greedy constructive method* is based on cluster first-route second principle: Customers are first clustered into feasible groups to be served by the same vehicle (cluster first) without regard to any preset ordering and then efficient routes are designed for each cluster (route second) (Simchi-Levi *et al*, 2005). It is made up of two parts:

(1) Partitioning $V\backslash\{0\}$ in K clusters, one for each route that should be designed, and
(2) designing a route for the locations belonging to each cluster and the depot.

The first part involves choosing the locations (orders) belonging to each cluster and is based on the following ideas: (a) approach our routing problem as a generalized assignment problem (GAP) as proposed by Fisher and Jaikumar (1981); (b) solve the GAP using as greedy function the one proposed by Martello and Toth (1990); this guide function evaluates, in each step, the benefit of assigning each not-assigned point.

More precisely, let us denote a_{ij} as the cost of assigning the location i to cluster j; $\forall\, i\in V\backslash\{0\}$; $\forall\, j=1,\ldots,K$.

This cost is calculated as proposed in Fisher and Jaikumar (1981). The following pseudocode describes this part.

Random Greedy Constructive Method – PART 1 (Clusters)

Initialize: Do $A=\varnothing$; $P=V\backslash\{0\}$
Repeat
(1) Calculate $\Delta i=a_{ij}{}^{**}(i)-a_{ij^*(i)}\quad \forall i\in P\backslash A$;
where
$j^*(i)=$ *argmin* $\{a_{ij}/$ *the assignment of i to j is feasible,* $j=1,\ldots,K\}$
$j^{**}(i)=$ *argmin* $\{a_{ij}/$ *the assignment of i to j is feasible,* $j=1,\ldots,K;\ j\neq j^*(i)\}$
(2) Calculate $\Delta_{min}=min\ \{\Delta_i/i\in P\backslash A\}$ *and* $\Delta_{max}=max\ \{\Delta_i/i\in P\backslash A\}$
(3) Build $RCL=\{i\in P\backslash A/\Delta_i\geqslant\alpha\ \Delta_{min}+(1-\alpha)\quad \Delta max\}$
(4) Choose $i^*\in$ *RCL randomly*
(5) Assign i^* *to* $j^*(i^*)$ *and do* $A=A\cup\{i^*\}$
until $A=P$

As can be observed, in every iteration the chosen location does not necessarily corresponds to the highest Δ_i, but a restricted list with the best candidates, *RCL* (corresponding to the highest Δ_i values) is built. Then, a location from this list is randomly chosen and assigned to its best cluster. The parameter α controls the level of randomization: the case $\alpha=0$, corresponds to a pure greedy algorithm (*RCL* is composed only by the location corresponding to $\Delta_{\max}$), while $\alpha=1$, is equivalent to a random construction ($RCL=P\backslash A$). A judicious selection of the value of α provides a balance between diversification and solution quality.

In order to favour the level of diversification, a modification has been added to this Part 1 of the random greedy constructive method. Specifically, let us define

freq(i,j) number of times that location i has been assigned to cluster j in previous executions of *Random Greedy Constructive Method*; $\forall\, i\in V\backslash\{0\};\ \forall\, j=1,\ldots,K$;

$$freq\_max(i)=\max\{freq(i,j)/\forall j=1,\ldots,K\}\quad \forall i\in V\backslash\{0\};$$
$$a\_max_i=\max\{a_{ij}/\forall j=1,\ldots,K\}\quad \forall i\in V\backslash\{0\};$$
$$a\_min_i=\min\{a_{ij}/\forall j=1,\ldots,K\}\quad \forall i\in V\backslash\{0\};$$

and

$$a'_{ij}=a_{ij}-\beta^*(freq(i,j)/freq\_max(i))^*(a\_max_i-a\_min_i),$$
$$\forall i\in V\backslash\{0\};\forall j=1,\ldots,K.$$

Now, in Part 1 of the *Random Greedy Constructive Method*, the step 1) is replaced by:

$$(1')\ \textit{Calculate}\ \Delta_i=a'_{ij^{**}(i)}-a'_{ij^*(i)}\quad \forall i\in P\backslash A;$$

where $j^*(i)=$ argmin $\{a'_{ij}/j=1,\ldots,K;$ the assignment of i to j is feasible$\}$ and $j^{**}(\mathrm{i})=$ argmin $\{\mathrm{a}'_{ij}/j=1,\ldots,K;$ $j\neq j^*(\mathrm{i})$ the assignment of i to j is feasible$\}$.

In this way, we have added penalizations to the cost coefficients, which help the method to make different assignments in repeated applications of the construction procedure. Not only the level of diversification increases but also we have observed that the quality of solutions may improve with a judicious choice of value of β.

Finally, part 2 of the random *greedy constructive method* consists of designing a route with the locations belonging to each cluster plus the depot. It is performed using the heuristic GENI (Gendreau *et al*, 1992), which is a well-known constructive method for the symmetric TSP. It has shown to be an efficient and effective method (Cordeau *et al*, 1997; Brandão, 2009). In this way, K routes are obtained.

3.2. Local search method

The solutions obtained by the *random greedy constructive method* (in Phase I) and by the *path generator method* (in Phase II, explained later in Section 3.3) are improved using a local search procedure. This method iteratively replaces the current solution by a better one in its neighbourhood and finishes when no better solution is found.

Let us denote $N(S)$ the neighbourhood of a solution S, and $f(S)$ its objective value (total distance). This local search method can be described in a pseudocode as follows.

Local Search Method

Read initial Solution S
Repeat
  (a) Make previous_value $= f(S)$
  (b) Search $f(S^*) = \min \{ f(S')/S' \in N(S) \}$
  (c) If $f(S^* < f(S)$ *then make* $S = S^*$
until $f(S^*) \geqslant$ *previous_value*

In our implementation $N(S)$ is obtained by performing the following three types of changes in S, which are illustrated in Figures 1, 2, and 3:

(a) Interchange two consecutive chains of points in a route (Or, 1976).
(b) Interchange two chains from two different routes (Taillard *et al*, 1997).
(c) Move a chain from a route to another different route.

In order to save computing time and to accelerate the execution of the local search method, we have applied a mechanism based on the geometric version of the 2-opt heuristic suggested by Bentley (1992) in the context of travelling salesman problem. This mechanism consists in dividing the neighbourhood into subsets of moves (or buckets). A binary variable is assigned to each sub-neighbourhood to indicate whether or not it is active. At the beginning, all the sub-neighbourhoods are active. In each iteration, the procedure explores only the active sub-neighbourhoods: if an active sub-neighbourhood fails to yield an improving solution, then the sub-neighbourhood becomes inactive for the next iteration. If an improving move is executed, then all affected sub-neighbourhoods become active.

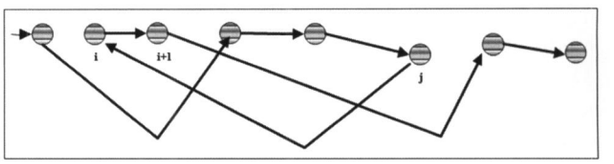

Figure 1 Interchange of two consecutive chains in the same route.

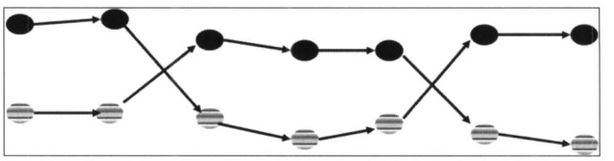

Figure 2 Interchange of two chains from different routes.

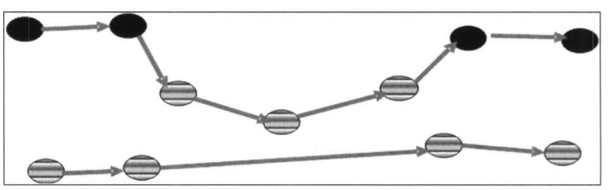

Figure 3 Movement of a chain from one route to other.

In our implementation, for each solution S we define the following sub-neighbourhoods:

$A(j)$ are solutions obtained by interchanging two consecutives chains in route j; $j = 1, \ldots, K$;
$B(j,j')$ are solutions obtained by interchanging two chains belonging respectively to routes j and j'; $j = 1, \ldots, K-1$ and $j' = j+1, \ldots, K$;
$C(j,j')$ are solutions resulting by moving a chain from route j to route j' $j, j' = 1, \ldots, K$ and $j' \neq j$.

We have empirically verified that using this scheme the local search method becomes between six and eight times faster.

3.3. Path generator method

In order to improve the solutions obtained in Phase I, the *path generator method* will be applied to every pair of solutions belonging to *Set_of_Best* (the set with the best solutions obtained in Phase I). This method generates

a *path* of solutions linking S_1 and S_2, with S_1, $S_2 \in Set\_of\_Best$. The local search method is applied then to all these new generated solutions.

Let us explain the way we designed this *path generator method*. First of all, solutions S_1 and S_2 will be represented as two big tours, assembling the routes and removing the depot 0.

With the purpose of illustration this, let us suppose that S_1 is made up of the routes:

$$0—1—4—2—3—0 \text{ and } 0—5—6—8—7—0;$$

Then, the associated big tour BT_1 will be

$$BT_1\text{: } 1—4—2—3—5—6—8—7.$$

In the same way, if S_2 is made up of the routes:

$$0—1—2—6—8—0 \text{ and } 0—5—4—3—7—0;$$

then, the associated big tour is

$$BT_2\text{: } 1—2—6—8—5—4—3—7.$$

Next, starting from BT_1, movements that introduce attributes contained in BT_2 are performed iteratively until BT_2 is obtained. In this way, a sequence BT^1, BT^2, BT^3 ... is created; in each iteration, the big tour is more similar to BT_2 than the previous one.

Finally, for each one of these intermediate big tours (BT^1, BT^2, BT^3 ...), a solution is formed by splitting the big tour into sub-chains that correspond to feasible routes.

There are, still, some questions that must be addressed before the method is well defined:

- What kinds of movements are permissible for obtaining intermediate big tours?
- How to select the appropriate movement to perform in each iteration?
- How to partition the intermediate big tours to create feasible solutions?

For obtaining the next intermediate big tour, interchanges between two consecutives chains of locations are checked in the current one. We will select and execute the first which results in a big tour that has more common arcs with BT_2 than the current one.

In order to illustrate this, we continue with the same example used previously. As before:

$$BT_1 = 1—4—2—3—5—6—8—7;$$
$$BT_2 = 1—2—6—8—5—4—3—7.$$

Then, the following sequence of intermediate big tours may be obtained:

$$BT^0 = BT_1 = 1—4—2—3—5—6—8—7;$$
$$BT^1 = 1—4—2—6—8—3—5—7;$$
$$BT^2 = 1—2—6—8—3—5—4—7;$$
$$BT^3 = BT_2 = 1—2—6—8—5—4—3—7.$$

As can be observed, BT^0 has in common with BT_2 only the arc (6, 8); BT^1 is obtained from BT^0 by interchanging 3—5 and 6—8. As a result, BT^1 has in common with BT_2 arcs (2, 6) and (6, 8). After interchanging 4 and 2—6—8—3—5 in BT^1, we obtain BT^2, which has in common with BT_2 the arcs (1, 2), (2, 6), (6, 8), and (5, 4). Finally, BT^3 is obtained from BT^2 by interchanging 3 and 5—4 and as a result, $BT^3 = BT_2$.

At last, in order to explain how the big tours are partitioned to create feasible solutions, let us consider the big tour

$$b_1—b_2—b_3— \ldots —b_n;$$

where b_i denotes the location that is in position i in this big tour. For simplicity, let us define $b_0 = 0$. The following directed graph $GR = (Nod, Arc)$ is created where $Nod = \{0, 1, \ldots, n\}$ is the set of nodes, and $Arc = \{(i, j)/i < j;\ i, j \in Nod\}$ the set of arcs.

Let us define $\forall\ (i,j) \in Arc$:

u_{ij} is the cost of the route that starts in 0, visits locations $b_{i+1}, b_{i+2}, \ldots, b_j$, and returns to 0, if this route is feasible; and $\propto$ otherwise.

Then, to find an optimal partition of the (directed) big tour into feasible routes, that is, routes with minimum total distance, a shortest path problem (SPP), Ahuja *et al* (1993) is solved from node 0 to n in the graph GR with arc costs u_{ij}.

For example, let us consider the big tour $BT^1 = 1—4—2—6—8—3—5—7$ and suppose that the solution of the SPP is 0—2—4—7. Then, the routes obtained, which make up solution S^1, are the following:

$$0—1—4—2—0;$$
$$0—6—8—0;$$
$$0—3—5—7—0.$$

In this way, a solution for every intermediate big tour is obtained, that is, a set of solutions S^1, S^2, S^3, ..., are obtained from big tours BT^1, BT^2, BT^3 .... The idea of considering the problem of finding the optimum split as a SPP was first proposed by Beasley (1983) and it has been used more recently by Prins (2004). The SPP is a polynomial problem and there exists classical and very well-known algorithms to solve it. Specifically, the Dijkstra method (Dijkstra, 1959) with a structure of binary heap (Beasley, 1983) has shown to work efficiently.

4. Computational experiments

Three sets of computational tests were conducted to check the performance of the proposed strategy. The first set compares our algorithm with commercial optimization

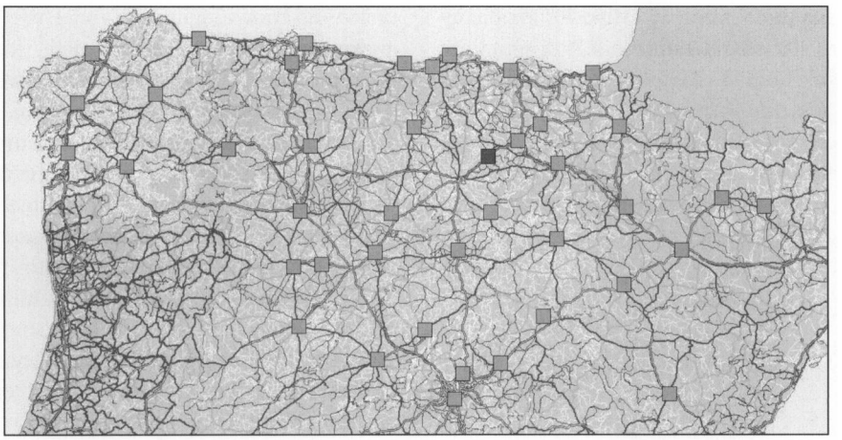

Figure 4 Locations of depot (in black) and distribution centres (in grey).

software (Cplex) in small instances. The second group of experiments solves real-data-based instances from the bakery company. As there are no other algorithms designed to this specific problem, we made the necessary modifications to treat the problem as a PVRP, in order to be able to compare our algorithm with the most recent and well-known algorithms for the PVRP. The aim of these two first sets of experiments is to assess the quality of the proposed algorithm.

Finally, the aim of using the third set of experiments is to check how the flexibility in delivery dates leads to a significant decrease in the total distance travelled over the planning horizon. As currently, the company performs day-to-day routing, these results show the convenience of considering flexibility in the delivery dates and solving the problem with the methodology proposed in this work. For this goal, we compare the results yielded by our algorithm for several instances, when $g=0$ ('not flexibility' or a day-to-day routing) and $g=1$ ('flexibility'). Real and pseudo-real instances were used in this set of experiments.

All the procedures have been implemented in object Pascal using Borland Delphi (ver 5.0) compiler and the algorithm has been run on a Corel 8400 PC with a 2.66 GHz CPU. The values of the parameters used are: $max\_iter=1000$, $n\_best=12$, $\alpha=0.5$, and $\beta=1$. These two last parameter values have been chosen after previous fine-tuning experiments. All instances used for the experiments are available for interested readers.

4.1. Comparison with commercial software

In this subsection, we compare our algorithm with the available commercial optimization software. Specifically, ILOG CPLEX 11.2 (www.ilog.com) is used for this comparison, running in a Sun Fire V440 Ultra Sparc III, 1062 GHz. Cplex uses the mixed-integer linear formulation stated in Section 2.

For this experiment we have considered small problem instances. Specifically, 45 instances have been randomly generated as follows:

- Three different orders have been considered $n=$ 10, 12, 15.
- For every value of n, five distance matrices are generated: every order is assigned randomly to a real distribution centre. Depot is assigned to real depot in Briviesca. Figure 4 shows the geographical locations of real distribution centres and depot. The distances are calculated as rounded Euclidean distances.
- For every distance matrix, the demands q_i are generated between 1 and 12 and the *deadline* e_i between 1 and 5.
- The value of Q and K are set to 30 and 2, respectively.
- For every distance matrix, three values of g ($g=1$, 2, 3) are considered.

For these small instances, only the Phase I (GRASP) of the algorithm is used, because, as it can be observed in Table 1, with this phase the optimum has been already achieved. In this table, Columns 4 and 6 (T.Dist) exhibit the total distance obtained by Cplex and GRASP, respectively, for each instance. Columns 5 and 7 (*Time*) show the computational time in seconds consumed by Cplex and GRASP, respectively. The last column (*T. to Best*) shows the spent time until the best solution is achieved by GRASP.

On the one hand we can see from Table 1 that our GRASP is able to find an optimal solution in very short computational time for all these small instances. For every instance, the method uses very short computational time (<5 s) in overall execution. Furthermore, our method finds the best solution spending <0.055 s in all instances (40 from the 45 instances spend <0.010 s). On the other hand, Cplex spends much more time than our method, in particular with higher values of n or g. Summarizing: these

Table 1 Comparison between Cplex and GRASP (first phase of the algorithm)

n	*No. of matrix*	*g*	*Cplex*		*GRASP*		
			T.Dist	*Time*	*T.Dist*	*Time*	*T. to best*
		1	2936	0.12	2936	0.802	0.003
	1	2	2877	0.24	2877	1.066	0.002
		3	2432	3.87	2432	1.209	0.001
		1	2318	0.46	2318	0.683	0.003
	2	2	2149	6.64	2149	1.102	0.002
		3	1947	11.18	1947	1.408	0.002
		1	2154	0.13	2154	0.727	0.003
10	3	2	2082	0.41	2082	1.026	0.007
		3	1771	4.38	1771	1.906	0.002
		1	2296	2.90	2296	1.32	0.002
	4	2	1826	11.15	1826	1.677	0.002
		3	1826	12.97	1826	1.808	0.002
		1	2283	0.18	2283	0.7	0.001
	5	2	2172	3.79	2172	0.81	0.001
		3	1889	8.08	1889	0.939	0.001
		1	2515	0.32	2515	0.911	0.003
	1	2	2164	25.84	2164	1.414	0.001
		3	2106	29.20	2106	1.977	0.002
		1	3200	0.60	3200	1.324	0.002
	2	2	2430	14.84	2430	2.053	0.017
		3	2323	13.27	2323	2.074	0.004
		1	3212	0.28	3212	0.832	0.005
12	3	2	2813	11.70	2813	1.473	0.002
		3	2584	11.92	2584	1.585	0.011
		1	2472	8.53	2472	1.507	0.004
	4	2	2209	17.04	2209	2.33	0.002
		3	2209	21.66	2209	2.886	0.003
		1	2432	3.13	2432	1.036	0.002
	5	2	2280	27.29	2280	1.811	0.01
		3	2102	100.12	2102	2.336	0.007
		1	3595	10.33	3595	1.869	0.001
	1	2	2949	464.80	2949	2.77	0.002
		3	2769	495.74	2769	3.567	0.021
		1	3661	18.42	3661	2.044	0.004
	2	2	3099	296.64	3099	2.537	0.002
		3	3092	682.87	3092	2.936	0.002
		1	3337	8.47	3337	1.407	0.001
15	3	2	2914	84.14	2914	2.247	0.006
		3	2759	748.87	2759	2.988	0.035
		1	3324	517.97	3324	2.313	0.055
	4	2	2713	2644.68	2713	3.771	0.004
		3	2713	3552.31	2713	4.272	0.004
		1	3117	0.50	3117	1.872	0.002
	5	2	2774	32.14	2774	3.176	0.006
		3	2309	44.64	2309	3.547	0.003

tests show that the commercial software had troubles to solve even small instances of this model, while GRASP only needs a few seconds to obtain an optimal solution.

4.2. Comparison with methods for PVRP

In the second set of computational experiments, our proposed algorithm is compared against recent and/or very well-known methods for PVRP from literature in instances similar to the real case. In PVRP, a planning period of T days is considered. Every customer has a frequency fr (ie, number of days the customer should be served) and a set of possible visiting schedules (every visiting schedule is a feasible set of fr days in which the customer can be served). The objective is to assign a visiting schedule to every client and to design the corresponding routes (one per vehicle), for every day of the planning period in order to minimize the total distance travelled.

For establishing a correspondence between the two models, note that the 'customers' can be identified as orders, $T = H$; for every customer $fr = 1$ and there is one visiting schedule for each feasible day.

Specifically, our algorithm is compared with the following methods for PVRP: The tabu search method proposed by Cordeau *et al* (1997), CGL, scatter search method proposed by Alegre *et al* (2007), ALP, and the VNS method proposed by Hemmelmayr *et al* (2009), HDH. For these experiments real-data-based instances have been generated as follows:

- The central depot is located in Briviesca, Burgos, where the company has its factory. There are 41 distribution centres located geographically in different cities of northern Spain. Figure 4 shows these locations, the central depot in black, and the distribution centres in grey. Usually, every distribution centre asks for one to three orders every week (on different days). Thus, the number of orders per week varies from 41 to 100 more or less (not all distribution centres ask for three orders the same week). For this reason, the following numbers of orders (n) have been considered $n = 41$, 62, 82, and 102. For every value of n, five instances are generated.
- Every order has assigned the corresponding x–y coordinates of its distribution centre. The distances are calculated as Euclidean distances in order to match the format used by the PVRP methods mentioned before.
- Demand q_i for each order is generated between 1 and 12.
- The vehicle capacity Q is set equal to 30.
- Maximum number of days g that an order could be served before its *deadline* is 1.

The CGL algorithm was run in a Dell Precision T7400 with four processors at 3.0 GHz CPU (a single processor was used to run the tests); the ALP algorithm was run on a Corel 8400 PC at 2.66 GHz CPU; the HDH algorithm was run on a 64 bits dual core PC, with 3.2 GHz CPU. Table 2 summarizes the results obtained for every method, specifically the total distance (*T.Dist*) and the total computational time in seconds (*T.Comp*). In addition, for our algorithm the total distance obtained in each phase (*T.Dist(I)* for GRASP phase and *T.Dist(II)* for PR phase) is shown. The best result for every instance is marked in bold.

Table 2 Comparison between our algorithm and PVRP methods in pseudo-real instances

Instance		*CGL*		*ALP*		*HDH*		*GRASP + PR*		
n	*No.*	*T.Dist*	*Time*	*T.Dist*	*Time*	*T.Dist*	*Time*	*T.Dist (I)*	*T.Dist (II)*	*Time*
41	1	**6425**	36	**6425**	33	**6425**	80	**6425**	**6425**	29
41	2	**6173**	37	6315	31	6177	81	**6173**	**6173**	28
41	3	**6356**	41	**6356**	33	**6356**	82	**6356**	**6356**	29
41	4	**6903**	35	**6903**	35	6922	84	**6903**	**6903**	26
41	5	**6833**	30	**6833**	31	**6833**	87	**6833**	**6833**	23
62	1	9298	58	9318	98	9336	166	9318	**9297**	81
62	2	**8765**	62	8824	92	8785	171	8851	**8765**	81
62	3	8833	58	8886	95	8849	177	**8814**	**8814**	71
62	4	**9089**	41	**9089**	77	**9089**	174	**9089**	**9089**	72
62	5	**8838**	68	8852	117	8878	163	8852	**8838**	96
82	1	**10779**	89	10913	206	10827	276	10941	10789	163
82	2	10460	93	10481	217	10501	283	10439	**10431**	203
82	3	10148	98	**10076**	210	10139	287	10087	**10076**	199
82	4	10555	72	**10550**	234	10620	289	10569	**10550**	177
82	5	10611	91	**10593**	217	10618	307	10705	10664	209
102	1	13313	112	13285	377	**13192**	434	13361	13204	349
102	2	13888	92	13835	389	13894	465	13984	**13828**	325
102	3	11921	117	11957	468	11981	455	11992	**11899**	389
102	4	12515	126	12424	454	12483	438	12557	**12355**	402
102	5	12591	127	12574	519	12640	439	12638	**12519**	416

From Table 2, it can be observed that the four algorithms obtain similar results in smallest instances ($n = 41$ and 62). In larger instances ($n = 82$ and 102), our GRASP–PR algorithm performs better than CGL, ALP, and HDH. Only in three from the 20 instances our GRAS–PR method performs worse than some other strategy. In the remaining 17 instances, our GRASP–PR obtains the best result. CGL obtains the best result in nine instances, ALP in eight instances and HDH in five. It must be pointed out that PR phase improves significantly the results obtained by GRASP phase in 13 instances; specifically it happens in the 10 largest instances (that is $n = 82$ and 102). On the other hand, our GRASP + PR algorithm, ALP, and HDH use similar amount of computational time while CGL spends less computational time in larger instances.

Table 3 below presents the deviation (%) of these obtained results from the best-known solution in every instance as well as the mean deviation. The best-known solution is obtained running algorithms ALP, HDH, and GRASP + PR with more computational time. From Table 3, it can be observed that our GRASP + PR algorithm is able to find the best-known solution in 13 out of 20 instances (more than the rest of the algorithms) and the mean deviation is $<0.18\%$ (less than other algorithms).

Note that we do not intend to rank algorithmic approaches, but to examine the opportunity of our proposed method for this specific model. For this goal, we compared our method with available tools (commercial and from literature). In this sense we should point out that the methods used for comparison (CGL, ALP, and HCH) have been designed for different types of instances of PVRP, while our method has been designed 'ad-hoc' for the specific model proposed and tackled in this work. The results indicate the convenience of using the method proposed in this paper.

4.3. Analysis of the influence of flexibility in delivery dates: comparison to the company's current solution

The purpose of the third set of computational experiments is to study the effect or influence that flexibility on delivery dates has over the total travelled distance, that is, we compare flexibility on delivery dates with the way the company is currently conducting its routing. The numerical results obtained by our algorithm when $g = 0$ ('not flexibility', company's current solution), and $g = 1$ ('flexibility', proposed solution) are compared solving some real instances and the pseudo-real instances already described in the previous subsection.

4.3.1. Results regarding real instances. This section shows the obtained results using real data provided by the company. The first eight instances correspond to the first weeks of 2009, while the last one corresponds to October 2002. For this last instance, the company provided the detailed data about orders, distribution centres, etc. However, for the 2009 instances, only the results shown in Table 4 might be published. Nevertheless

Table 3 Percentage deviation from best-known solutions

Instance		*CGL*	*ALP*	*HDH*	*GRASP–PR*	*Best-known solution*
n	*No.*	*Percentage deviation from best-known solution*				
41	1	0	0	0	0	**6425**
41	2	0	2.30	0.06	0	**6173**
41	3	0	0	0	0	**6356**
41	4	0	0	0.28	0	**6903**
41	5	0	0	0	0	**6833**
62	1	0.01	0.23	0.42	0	**9297**
62	2	0	0.67	0.23	0	**8765**
62	3	0.22	0.82	0.40	0	**8814**
62	4	0	0	0	0	**9089**
62	5	0	0.16	0.45	0	**8838**
82	1	0.57	1.82	1.02	0.66	**10718**
82	2	0.39	0.59	0.79	0.11	**10419**
82	3	0.71	0	0.63	0	**10076**
82	4	0.05	0	0.66	0	**10550**
82	5	0.47	0.30	0.54	0.97	**10561**
102	1	1.77	1.55	0.84	0.93	**13082**
102	2	0.62	0.24	0.67	0.19	**13802**
102	3	0.18	0.49	0.69	0	**11899**
102	4	1.71	0.98	1.46	0.41	**12304**
102	5	0.86	0.72	1.24	0.28	**12484**
Mean		**0.378**	**0.543**	**0.519**	**0.177**	

Table 4 Comparison against company's current solution obtained for the real instances

Instance	*Total distance*		
n	$g=0$	$g=1$	*Per cent reduction*
41	8111	6406	21.02
50	9417	7633	18.94
57	10015	8406	16.07
62	11356	9256	18.49
70	11618	9377	19.29
75	11813	9460	19.92
86	14174	11406	19.53
90	13765	11071	19.57

we believe that they are representative of the effect and impact of the designed approach for the company.

Table 4 shows for 2009 instances the number of orders (n), the total distance without flexibility ($g=0$), the total distance with flexibility ($g=1$), and the achieved reduction (*% reduction*).

Notice that allowing flexibility, in the delivery dates in these instances, reduces the total travelling cost in about 20%.

Next we analyse the instance corresponding to October 2002. This instance, with a small number of orders, does not represent the current situation and, as can be seen below, shows worse results than those obtained on real recent instances. However, just because the data are old enough, we can use them here as illustration.

Table 5 Orders in the distribution centres for October 2002 instance

	Monday (M)	*Tuesday (T)*	*Wednesday (W)*	*Thursday (H)*	*Friday (F)*
Asturias	18	18	18	1	18
León	5	4	4	3	5
Ponferrada	6	—	7	7	9
Coruña	17	—	12	20	18
Lugo	9	11	14	6	9
Santiago	4	4	3	4	4
Pontevedra	22	21	20	22	—
Orense	6	8	9	10	—

At that time the company had the distribution centres shown in Table 5. The number of orders per week by each of them was between 4 and 5. Table 5 shows the orders (in pallets) of each distribution centre and the Figure 5 shows the geographical locations with the corresponding names.

The results obtained for $g=0$ and 1 are shown in Table 6 and 7, respectively.

Note that the total distance travelled for $g=0$ is 14 502 and for $g=1$ is 12 911, resulting in a reduction around of 11% in this small instance. This confirms that allowing some flexibility in the delivery date leads to significant reductions in total travelling costs.

4.3.2. Results regarding pseudo-real instances. Table 8 shows the results obtained without flexibility ($g=0$) and

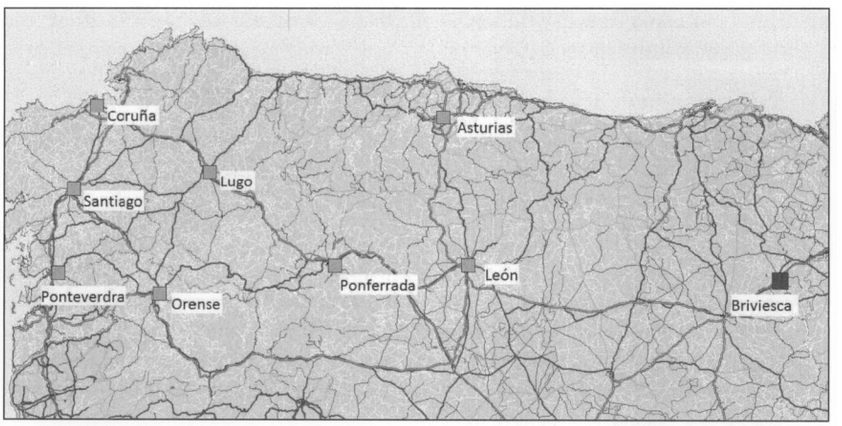

Figure 5 Locations of depot (in black) and distribution centres (in grey).

Table 6 Routes and distance travelled by each route for October 2002 instance with $g=0$

Route no.	*Distance*	*Route*
1	789	Briviesca—Asturias (M)—Ponferrada (M)—León (M)—Briviesca
2	1115	Briviesca—Lugo (M)—Coruña (M)—Santiago (M)—Briviesca
3	1105	Briviesca—Pontevedra (M)—Orense (M)—Briviesca
4	1145	Briviesca—León (T)—Santiago (T)—Pontevedra (T)—Briviesca
5	990	Briviesca—Orense (T)—Lugo (T)—Briviesca
6	592	Briviesca—Asturias (T)—Briviesca
7	1115	Briviesca—Santiago (W)—Coruña (W)—Lugo (W)—Briviesca
8	789	Briviesca—León (W)—Ponferrada (W)—Asturias (W)—Briviesca
9	1105	Briviesca—Pontevedra (W)—Orense (W)—Briviesca
10	939	Briviesca—Ponferrada (X)—Orense (X)—Briviesca
11	1101	Briviesca—Asturias (X)—Coruña (X)—Lugo (X)—Briviesca
12	1145	Briviesca—León (X)—Santiago (X)—Pontevedra (X)—Briviesca
13	592	Briviesca—Asturias (F)—Briviesca
14	1110	Briviesca—Santiago (F)—Coruña (F)—Briviesca
15	870	Briviesca—Lugo (F)—Ponferrada (F)—León (F)—Briviesca

Table 7 Routes and distance travelled by each route for October 2002 instance with $g=1$

Route no.	*Distance*	*Route*
1	1115	Briviesca—Santiago (M)—Coruña (M)—Lugo (M)—Briviesca
2	789	Briviesca—Asturias (T)—Ponferrada (M)—León (M)—Briviesca
3	1105	Briviesca—Orense (M)—Pontevedra (M)—Briviesca
4	943	Briviesca—Lugo (T)—Asturias (M)—Briviesca
5	1105	Briviesca—Pontevedra (W)—Orense (T)—Briviesca
6	1117	Briviesca—León (T)—Santiago (T)—Santiago (W)—Coruña (W)—Ponferrada (W)—Briviesca
7	1105	Briviesca—Orense (W)—Pontevedra (T)—Briviesca
8	990	Briviesca—Lugo (X)—Lugo (W)—Orense (X)—Briviesca
9	789	Briviesca—León (W)—Ponferrada (X)—Asturias (W)—Briviesca
10	1041	Briviesca—Lugo (F)—Coruña (X)—Briviesca
11	1141	Briviesca—Santiago (X)—Santiago (F)—Pontevedra (X)—Briviesca
12	632	Briviesca—Asturias (X)—Asturias (F)—León (X)—León (F)—Briviesca
13	1039	Briviesca—Coruña (F)—Ponferrada (F)—Briviesca

with flexibility ($g=1$) using the pseudo-real instances described in Subsection 4.2.

As before, significant reductions are obtained when flexibility in delivery dates is allowed. This directly impacts upon the economy of the company, so it is recommendable.

5. Concluding remarks

In this paper we addressed a particular real VRP presented to the authors. Even though this problem has been proposed by a bakery company, the same problem or a similar one can be found in other delivering companies. In order to reduce the total distance travelled by the vehicles over the planning horizon, some flexibility in the delivery dates has been considered. However, the current approach at the company cannot handle the additional flexibility. As the computational experiments verified, this flexibility allows a reduction of about 20% in the travel cost in real-data-based instances, which represents a very significant decrease in the transportation costs for the company.

A formulation as a mixed-integer linear model for this new problem is proposed. In addition, we have designed and implemented a two-phase method based in GRASP

Table 8 Comparison against company's current solution obtained for the pseudo-real instances

Instance		*Total distance*		
n	*No.*	$g=0$	$g=1$	*Per cent reduction*
41	1	8128	6425	20.95
41	2	7935	6173	22.20
41	3	8190	6356	22.40
41	4	8784	6903	21.41
41	5	8355	6833	18.22
62	1	11 590	9297	19.78
62	2	10 538	8765	16.82
62	3	10 855	8814	18.80
62	4	11 214	9089	18.95
62	5	11 323	8838	21.95
82	1	12 980	10 789	16.88
82	2	12 683	10 431	17.76
82	3	12 725	10 076	20.81
82	4	13 561	10 550	22.20
82	5	13 357	10 664	20.16
102	1	16 016	13 204	17.56
102	2	16 997	13 828	18.64
102	3	14 839	11 899	19.81
102	4	15 097	12 355	18.16
102	5	15 166	12 519	17.45

and path-relinking metaheuristic strategies for this model. For small instances, our method obtains optimal solutions in very short computational time (<0.01 s), while commercial optimization software (Cplex) needs much more time, specifically 221 s in average, varying from 0.12 to 3500 s.

For pseudo-real and larger instances, our algorithm performs better than well known and recent methods for PVRP. This is concluded, first, from the fact that our algorithm reaches the best result in 17 of 20 instances analysed, which compares favourably with the nine, eight, and five best results obtained, respectively, by the other methods. Moreover, the average deviation from the best-known solution is $<0.18\%$, against 0.378, 0.543, and 0.519 obtained, respectively, by the other methods. In the largest instances ($n=82$ and 102), which are more like those found in practice, the better performance of our algorithm is more evident; it obtains seven best results out of 10 instances against the one, three, one best results obtained, respectively, by the other methods. For those instances, where large quantities are handled, any improvement is relevant. All this suggests the convenience of using our method that has been designed 'ad-hoc' for these kinds of problems.

An interesting direction of research in the near future is to study the problem from the perspective of the multi-objective optimization, taking into account two objectives: minimize the total distance travelled and minimize the cost of the stock generated when orders are delivered before its deadline.

Acknowledgements—We wish to thank sincerely J.F. Cordeau and V. Hemmelmayr for running our instances with their algorithms, which allowed the comparisons presented in this work. This work has been done when the last three authors enjoyed the hospitality of Applied Economic Department, University of Burgos, Spain. We would like to thank all colleagues from this department. This work was partially supported by CONACYT México (grants 61903, 61343), FEDER founds and Spanish Ministry of Science (project ECO2008-06159/ECON), Regional Government of 'Castilla y León' (project BU008A10-2) University of Burgos and *CajaBurgos* (internal projects 2009) and the Universidad Autónoma de Coahuila (México). These supports are gratefully acknowledged. Finally, thanks are due to the two anonymous referees who greatly helped to improve the presentation of this work.

References

Ahuja R, Magnanti T and Orlin J (1993). *Network Flows: Theory, Algorithms and Applications*. Prentice Hall: Englewood Cliffs, NJ.

Alegre J, Laguna M and Pacheco J (2007). Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts. *Eur J Opl Res* **179**: 736–746.

Beasley J (1983). Route-first cluster-second methods for vehicle routing. *Omega* **11**: 403–408.

Bentley J (1992). Fast algorithms for geometric traveling salesman problems. *ORSA J Comput* **4**: 387–411.

Bolduc M, Renaud J, Boctor F and Laporte G (2008). A perturbation metaheuristic for the vehicle routing problem with private fleet and common carriers. *J Opns Res Soc* **59**(6): 776–787.

Brandão J (2009). A deterministic tabu search algorithm for the fleet size and mix vehicle routing problem. *Eur J Opl Res* **195**(3): 716–728.

Bräysy O (2003). A reactive variable neighborhood search algorithm for the vehicle routing problem with time windows. *INFORMS J Comput* **15**(4): 347–368.

Bräysy O and Gendreau M (2005). Vehicle routing problem with time windows, part I: Route construction and local search algorithms. *Transport Sci* **39**(1): 104–118.

Christofides N and Beasley J (1984). The period routing problem. *Networks* **14**: 237–256.

Clarke G and Wright J (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Opns Res* **12**(4): 568–581.

Cordeau J, Gendreau M and Laporte G (1997). A tabu search heuristic for periodic and multidepot vehicle routing problems. *Networks* **30**: 105–119.

Cordeau J, Laporte G and Mercier A (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *J Opl Res Soc* **52**: 928–936.

Cordeau J, Laporte G, Savelsbergh P and Vigo D (2007). Vehicle routing. In: Barnhart C and Laporte G (eds). *Transportation, Handbooks in Operations Research and Management Science*. Elsevier: Amsterdam, pp 367–428.

Dijkstra E (1959). A note on two problems in connexion with graphs. *Numer Math* **1**: 269–271.

Dullaert W, Janssens G, Sorensen K and Vernimmen B (2002). New heuristics for the fleet size and mix vehicle routing problem with time windows. *J Opl Res Soc* **53**: 1232–1238.

Feo T and Resende M (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Opns Res Lett* **8**: 67–71.

Feo T and Resende M (1995). Greedy randomized adaptive search procedures. *J Global Optim* **6**: 109–133.

Fisher M (1994). Optimal solution of vehicle routing problems using minimum K-trees. *Opns Res* **42**: 626–642.

Fisher M and Jaikumar R (1981). A generalized assignment heuristic for vehicle routing. *Networks* **11**: 109–124.

Flisberg P, Bertil Liden B and Ronnqvist M (2009). A hybrid method based on linear programming and tabu search for routing of logging trucks. *Comput Opns Res* **36**: 1122–1144.

Francis P, Smilowitz K and Tzur M (2008). The period vehicle routing problem and its extensions. In: Golden B, Raghavan S and Wasil E (eds). *The Vehicle Routing Problem Latest Advances and New Challenges*. Springer: New York, pp 73–102.

Gajpal Y and Abad P (2010). Saving-based algorithms for vehicle routing problem with simultaneous pickup and delivery. *J Opl Res Soc* **61**: 1498–1509.

Gambardella L, Taillard E and Agazzi G (1999). MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In: Corne D, Dorigo M and Glover F (eds). *New Ideas in Optimization*. McGraw-Hill: New York.

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

Gendreau M, Hertz A and Laporte G (1994). A tabu search heuristic for the vehicle routing problem. *Mngt Sci* **40**: 1276–1290.

Gillet B and Miller L (1974). A heuristic algorithm for the vehicle-dispatch problem. *Opns Res* **22**: 340–349.

Glover F, Laguna M and Martí R (2000). Fundamentals of scatter search and path relinking. *Control Cybern* **39**: 653–684.

Goel A and Gruhn V (2008). A general vehicle routing problem. *Eur J Opl Res* **191**(3): 650–660.

Haghani A and Jung S (2005). A dynamic vehicle routing problem with time-dependent travel times. *Comput Opns Res* **32**(11): 2959–2986.

Hemmelmayr VC, Doerner KF and Hartl RF (2009). A variable neighbourhood search heuristic for periodic routing problems. *Eur J Opl Res* **195**: 791–802.

Imram A, Said Salhi S and Wassan N (2009). A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem. *Eur J Opl Res* **197**(2): 509–518.

Jozefowiez N, Glover F and Laguna M (2008). Multi-objective meta-heuristics for the traveling salesman problem with profits. *J Math Model Algor* **7**: 177–195.

Kontoravdis G and Bard J (1995). A GRASP for the vehicle routing problem with time windows. *ORSA J Comput* **7**: 10–23.

Li FY, Golden B and Wasil E (2007). The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Comput Opns Res* **34**(10): 2918–2930.

Marinakis Y, Migdalas A and Pardalos P (2007). A new bilevel formulation for the vehicle routing problem and a solution method using a genetic algorithm. *J Global Optim* **38**(4): 555–580.

Martello S and Toth P (1990). *Knapsack Problems Algorithms and Computer Implementations*. John Wiley & Sons: New York.

Mester D and Bräysy O (2005). Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Comput Opns Res* **32**(6): 1593–1614.

Or I (1976). *Traveling Salesman Type Combinatorial Problems and Their Relations to the Logistics of Blood Banking*. Dissertation, Northwestern University.

Osman I (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Ann Opl Res* **41**: 421–451.

Pisinger D and Ropke S (2007). A general heuristic for vehicle routing problems. *Comput Opns Res* **34**(8): 2403–2435.

Pitsoulis L and Resende M (2002). Greedy randomized adaptive search procedures. In: Pardalos P and Resende M (eds). *Handbook of Applied Optimization*. Oxford University Press: Oxford, pp 168–183.

Potvin J and Bengio S (1994). *A genetic approach to the vehicle routing problem with time windows*. Technical Report CRT-953, Centre de Recherche sur les Transports, Université de Montréal.

Potvin J and Naud M (2011). Tabu search with ejection chains for the vehicle routing problem with private fleet and common carrier. *J Opl Res Soc* **62**(2): 326–336.

Prins C (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Comput Opns Res* **31**: 1985–2002.

Prins C and Prodhon C (2007). Solving the capacitated location-routing problem by a cooperative Lagrangean relaxation-granular tabu search heuristic. *Transport Sci* **41**: 470–483.

Rochat Y and Taillard E (1995). Probabilistic diversification and intensification in local search for vehicle routing. *J Heuristic* **40**: 147–167.

Ropke S and Pisinger D (2006). A unified heuristic for a large class of vehicle routing problems with backhauls. *Eur J Opl Res* **171**(3): 750–775.

Simchi-Levi D, Chen X and Bramel J (2005). *The Logic of Logistics. Theory, Algorithms and Applications for Logistics and Supply Chain Management*. Springer: Berlin.

Taillard E, Badeau P, Gendreau M and Potvin J (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transport Sci* **31**: 170–186.

Tan KC, Chew YH and Lee LH (2006). A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. *Eur J Opl Res* **172**(3): 855–885.

Tarantilis C and Kiranoudis C (2007). A flexible adaptive memory-based algorithm for real-life transportation operations: Two case studies from dairy and construction sector. *Eur J Opl Res* **179**(3): 806–822.

Thangiah S, Osman I and Sun T (1994). *Hybrid genetic algorithm simulated annealing and tabu search methods for vehicle routing problem with time windows*. Technical Report 27, Computer Science Department, Slippery Rock University.

Toth P and Vigo D (eds) (2002). Monographs on discrete mathematics and applications. *Vehicle Routing Probl*. SIAM: Philadelphia, PA.

Vondouris C and Tsang E (1999). Guided local search and its application to the travelling salesman problem. *Eur J Opns Res* **113**: 469–499.

Zachariadis E, Tarantilis C and Kiranoudis C (2009a). A guided tabu search for the vehicle routing problem with two-dimensional loading constraints. *Eur J Opl Res* **195**(3): 729–743.

Zachariadis E, Tarantilis C and Kiranoudis C (2009b). A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Syst Appl* **36**(2): 1070–1081.

Received March 2010;
accepted March 2011 after one revision