Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 178 (2007) 755–766
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

An exact algorithm for a single-vehicle routing problem with time windows and multiple routes

Nabila Azi, Michel Gendreau, Jean-Yves Potvin *

Département d'informatique et de recherche opérationnelle and Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Québec, Canada H3C 3J7

Received 19 January 2005; accepted 8 February 2006
Available online 3 May 2006

Abstract

This paper describes an exact algorithm for solving a problem where the same vehicle performs several routes to serve a set of customers with time windows. The motivation comes from the home delivery of perishable goods, where vehicle routes are short and must be combined to form a working day. A method based on an elementary shortest path algorithm with resource constraints is proposed to solve this problem. The method is divided into two phases: in the first phase, all non-dominated feasible routes are generated; in the second phase, some routes are selected and sequenced to form the vehicle workday. Computational results are reported on Euclidean problems derived from benchmark instances of the classical vehicle routing problem with time windows.

Keywords: Transportation; Routing; Single vehicle; Time windows; Multiple uses; Elementary shortest paths

1. Introduction

In this work, we consider a variant of the vehicle routing problem with time windows (VRPTW) where the same vehicle can perform several routes during its workday. Surprisingly, this problem has received little attention in the literature in spite of its importance in practice. For example, in the home delivery of perishable goods, like food, routes are of short duration and must be combined to form a complete workday. We believe that this type of problem will become increasingly important in the future with the advent of electronic services, like e-groceries, where customers can order goods through the internet and have them delivered at home.

The vehicle routing problem with multiple uses of vehicles, but no time windows, has been addressed through heuristic means in [7,14]. In [14], different solutions to the classical vehicle routing problem are generated using a tabu search heuristic. The routes obtained are then combined to produce workdays for the vehicles by solving a bin packing problem, an idea previously introduced in [7]. A recent work in [3] reports about insertion heuristics that can efficiently handle different types of constraints including time windows and multiple uses of vehicles. In [2], the authors introduce the home delivery problem, which is more closely related to

* Corresponding author. Tel.: +1 514 343 7093; fax: +1 514 343 7121.
E-mail address: potvin@iro.umontreal.ca (J.-Y. Potvin).

0377-2217/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2006.02.019

real-world applications. Here, a probability of occurrence and a revenue are associated with each potential customer. When a new request occurs, a decision to accept or reject it must be taken in real-time, and a time window for service is determined. Although vehicle routes are generated and used to decide about the acceptance or rejection of a particular request, the "real" routes are executed later. Logistics and socio-economic considerations about different types of home delivery problems, with a particular emphasis on electronic groceries, can also be found in [8–11,15].

In this paper, an exact algorithm for solving a single-vehicle routing problem with time windows and multiple routes is reported. To the best of our knowledge, this is the first time that an exact algorithm is devised for this kind of problem. The outline of the paper is as follows. In Section 2, a mathematical programming formulation is proposed. The problem-solving approach, based on an elementary shortest path algorithm with resource constraints, is then presented in Section 3. Computational results on problem instances derived from Solomon's VRPTW testbed [13] are reported in Section 4. Finally, concluding remarks follow in Section 5.

2. Problem formulation

The problem considered can be stated as follows. We have a single vehicle of capacity Q delivering perishable goods from a depot to a set of customer nodes $N = \{1,2,\ldots,n\}$ in a complete directed graph with arc set A. A distance d_{ij} and a travel time t_{ij} are associated with every arc $(i,j) \in A$. Each customer $i \in N$ is characterized by a demand q_i, a service or dwell time s_i and a time window $[a_i, b_i]$, where a_i is the earliest time to begin service and b_i the latest time. Accordingly, the vehicle must wait if it arrives at customer i before time a_i. The vehicle workday is made of a set of routes $K = \{1,2,\ldots,k\}$ where each route starts and ends at the depot (some of these routes might be empty). We assume, without loss of generality, that the routes are served in the order $1,2,\ldots,k$. The depot is denoted by 0 or $n+1$ depending if it is the initial or terminal node of an arc, with $s_0 = s_{n+1} = 0$, $q_0 = q_{n+1} = 0$, $a_0 = a_{n+1} = 0$; $b_0 = b_{n+1} = \infty$; the symbol N^+ is used for $N \cup \{0, n+1\}$ and A^+ for $A \cup \{(0,n+1)\}$, where $(0,n+1)$ is a dummy arc with distance $d_{0,n+1} = 0$ and travel time $t_{0,n+1} = 0$. Due to the transportation of perishable goods, every customer in a route must be served before a given deadline associated with that route. The latter is defined by adding a constant $t_{\max}$ to the route start time. Also, a setup time σ^r for loading the vehicle is associated with each route $r \in K$. The objective is to minimize the total distance traveled to serve all customers while satisfying the capacity, time window and deadline constraints.

This problem can be formulated as follows, using M as an arbitrary large constant:

$$\min \quad \sum_{r\in K}\sum_{(i,j)\in A} d_{ij}x_{ij}^r \tag{1}$$

$$\text{s.t.} \quad \sum_{j\in N^+} x_{ij}^r = y_i^r, \quad i \in N, \quad r \in K, \tag{2}$$

$$\sum_{r\in K} y_i^r = 1, \quad i \in N, \tag{3}$$

$$\sum_{i\in N^+} x_{ih}^r - \sum_{j\in N^+} x_{hj}^r = 0, \quad h \in N, \quad r \in K, \tag{4}$$

$$\sum_{i\in N^+} x_{0i}^r = 1, \quad r \in K, \tag{5}$$

$$\sum_{i\in N^+} x_{i(n+1)}^r = 1, \quad r \in K, \tag{6}$$

$$\sum_{i\in N} q_i y_i^r \leqslant Q, \quad r \in K, \tag{7}$$

$$t_i^r + s_i + t_{ij} - M\left(1 - x_{ij}^r\right) \leqslant t_j^r, \quad (i,j) \in A^+, \quad r \in K, \tag{8}$$

$$a_i y_i^r \leqslant t_i^r \leqslant b_i y_i^r, \quad i \in N, \quad r \in K, \tag{9}$$

$$t_0^1 \geqslant \sigma^1, \tag{10}$$

$$t_{n+1}^r + \sigma^{r+1} \leqslant t_0^{r+1}, \quad r = 1,\ldots,k-1, \tag{11}$$

$$\sigma^r = \beta \sum_{i\in N} s_i y_i^r, \quad r \in K, \tag{12}$$

$$t_i^r \leqslant t_0^r + t_{\max}, \quad i \in N, \quad r \in K, \tag{13}$$

$$x_{ij}^r \text{ binary}, \quad (i,j) \in A^+, \quad r \in K, \tag{14}$$

$$y_i^r \text{ binary}, \quad i \in N, \quad r \in K, \tag{15}$$

where

- x_{ij}^r is 1 if arc $(i,j) \in A^+$ is in route r, 0 otherwise; note that $x_{0,n+1}^r$ is 1 if route r is empty;
- y_i^r is 1 if customer i is in route r, 0 otherwise;
- t_i^r is the time of beginning of service at customer i in route r;
- t_0^r is the start time of route r;
- t_{n+1}^r is the end time of route r.

In this formulation, Eq. (3) states that every customer should be visited exactly once. Eqs. (4)–(6) are flow conservation constraints that describe the

vehicle path. Eq. (7) states that the total demand on a route should not exceed the vehicle capacity. Eqs. (8)–(11) ensure feasibility of the time schedule. Eq. (12) defines the vehicle setup time as the sum of service times of all customers in a route, multiplied by parameter β. Finally, Eq. (13) corresponds to the deadline constraint for serving a customer. Note that Eq. (9) forces the t_i^r variables to 0 when customer i is not in route r. Consequently, Eq. (13) is automatically satisfied in this case.

In practice, it might not be possible to serve all customers with the vehicle due to the time window constraints. An alternative objective is thus to maximize the number of served customers and, for the same number of served customers, to minimize the total distance. The objective in the above formulation can be modified to account for the number of served customers, namely:

$$\min \sum_{r\in K}\sum_{(i,j)\in A} d_{ij}x_{ij}^r - \alpha \sum_{r\in K}\sum_{i\in N} y_i^r. \tag{16}$$

By setting the weighting parameter α to a sufficiently large value, the desired hierarchic objective is obtained. Note also that the equality sign in constraint (3) should be replaced by a $\leqslant$ sign. The latter problem is the one that is considered in the following.

3. Problem-solving approach

The problem is addressed via an approach that exploits an elementary shortest path algorithm with resource constraints, noted FDGG in the following [6]. The latter extends Desrochers' algorithm [5] for the shortest path problem with resource constraints to generate elementary paths only. After briefly introducing FDGG, our problem-solving methodology is described in the next subsections.

3.1. An elementary shortest path algorithm with resource constraints

FDGG is a label correcting algorithm that solves the elementary shortest path problem with resource constraints on graphs with, possibly, negative cycles. In this context, a path is characterized by the consumption of each resource, in addition to its length. Accordingly, when different paths lead to the same node, it might well be that no path dominates, or is better than the others, over all criteria. As a consequence, many different labels are typically maintained at each node (i.e., all non-dominated paths leading to that node).

Since elementary paths must be generated, cycles are detected by keeping a trace of previously visited nodes. More precisely, a path p from some origin node o to some node j is labeled with $R_p = (d_p, t_p^1, \ldots, t_p^l, s_p, V_p^1, \ldots, V_p^n)$, where $L = \{1, \ldots, l\}$ is the set of resources, d_p is the length of path p, t_p^k is the consumption of resource $k = 1, \ldots, l$, s_p is the number of unreachable nodes (either because they have already been visited or because their inclusion would violate one or more resource constraints) and $V_p^i = 1$ if node i is unreachable, 0 otherwise. The following dominance relation is then defined:

Dominance relation. If p and p' are two different paths from origin o to node j with labels R_p and $R_{p'}$, respectively, then path p dominates p' if and only if $d_p \leqslant d_{p'}$, $s_p \leqslant s_{p'}$, $t_p^k \leqslant t_{p'}^k$, $k = 1, \ldots, l$, $V_p^i \leqslant V_{p'}^i$, $i = 1, \ldots, n$.

That is, path p dominates p' if (1) it is not longer, (2) it does not consume more resources for every resource considered and (3) every unreachable node is also unreachable for path p'. Note that s_p, the number of unreachable nodes, is included in the label only to speed up the computations. As stated in [6], by eliminating paths through this dominance relation, only labels corresponding to non-dominated elementary paths are kept and a solution to the problem is obtained at the end.

Our problem-solving approach, based on this algorithm, is divided into two phases. In the first phase, feasible routes are constructed. Then, some of these routes are combined to form a workday for the vehicle. Through this two-phase approach, the routing issues are decoupled from the scheduling issues related to the generation of a workday. That is, when a route is known, the sum of the service times and the setup time are known. With the setup times, the coupling of different routes can then be done with standard time window constraints. This is explained in the following.

3.2. Phase 1

First, we want to generate a set of feasible routes that will be used as "building blocks" in the second phase. As the deadline constraint is quite restrictive in practice, even the largest feasible routes will contain only a few customers and a pure enumerative approach is viable. Through a suitable adaptation of the FDGG algorithm, however, it is possible to discard partial routes that are of no interest due to the availability of better alternative routes.

First, the distance d_{ij} on each arc is replaced by $d_{ij} - \alpha$, with $\alpha > \max_{(i,j)\in A} d_{ij}$. As it is always beneficial to extend a path with a new arc, the corresponding distance being negative, this modification allows the FDGG algorithm to generate paths that start and end at the depot (otherwise, it would be optimal to stay at the depot). Second, as our goal is not to find shortest routes, but rather to generate all feasible non-dominated routes, condition (3) in the FDGG dominance relation is modified and now states that the nodes visited in one path should be the same as those visited in the other path. Using the notation $W_p^i = 1$ if node i is in path p, 0 otherwise, the dominance relation is now defined as follows:

Dominance relation (modified). If p and p' are two different paths from origin o to node j with labels R_p and $R_{p'}$, respectively, then path p dominates p' if and only if $d_p \leqslant d_{p'}$, $t_p^k \leqslant t_{p'}^k$, $k = 1, \ldots, l$, $W_p^i = W_{p'}^i$, $i = 1, \ldots, n$.

A path p thus dominates another path p' if they both lead to the same node, going through the same intermediary nodes (although in a different order), and p is not longer and does not consume more resources than p', for every resource considered. In this application, the resource constraints are the time windows and route deadlines. Clearly, there is no interest in extending dominated paths, because we always want to visit a given subset of customers in the most efficient way. All feasible non-dominated closed routes obtained with this modified FDGG algorithm are kept and used in Phase 2 to generate a solution to our problem.

3.3. Phase 2

The FDGG algorithm is used again in Phase 2 to create a workday for the vehicle. Here, the original algorithm is applied on a transformed graph where the nodes correspond to the routes generated in Phase 1, plus two artificial nodes that correspond to the start and end of the vehicle workday. In the graph, there is an arc between nodes r and r' if (1) the two subsets of customers in routes r and r' are disjoint and (2) it is feasible to serve route r' after route r, where feasibility is determined through departure time windows (see Subsections 3.3.1 and 3.3.2). It should be noted that after applying this graph reduction procedure, it is still possible to have paths that connect routes with non-disjoint subsets of customers (only *consecutive* routes with non-disjoint subsets of customers are eliminated at this point). A single resource unit is thus associated with each customer. When the FDGG algorithm is applied on the transformed graph and some route node r' is visited, the resource unit of every customer in r' is automatically consumed.

The cost on arc (r, r') in the transformed graph relates to the objective in Eq. (16) for route r' and is set to the length of route r' minus the weighting parameter α times the number of customers in route r'. That is,

$$\sum_{(i,j)\in A} d_{ij} x_{ij}^{r'} - \alpha \sum_{i\in N} y_i^{r'}, \tag{17}$$

with $\alpha > 2\max_{i\in N}\{\max\{d_{0i}, d_{i(n+1)}\}\}$. With this requirement on parameter α, and assuming that the triangular inequality holds, the costs are all negative. Namely,

$$\begin{aligned}
\sum_{(i,j)\in A} d_{ij} x_{ij}^{r'} &\leqslant \sum_{i\in N} (d_{0i} + d_{i(n+1)}) y_i^{r'} \quad \text{(triangular inequality)} \\
&\leqslant \sum_{i\in N} 2\max\{d_{0i}, d_{i(n+1)}\} y_i^{r'} \\
&\leqslant 2\max_{i\in N}\{\max\{d_{0i}, d_{i(n+1)}\}\} \sum_{i\in N} y_i^{r'} \\
&< \alpha \sum_{i\in N} y_i^{r'}.
\end{aligned}$$

Thus, it is always beneficial to serve an additional customer if it is feasible to do so. In the transformed graph, there is also an arc from the artificial start node to every route node and from every route node to the artificial end node (in the latter case, the costs are set to 0).

As previously mentioned, time windows are associated with route nodes. These are calculated as follows.

3.3.1. Latest departure and arrival times

Let us assume that route r corresponds to the sequence $(0 = i_0, i_1, i_2, \ldots, i_{n_r}, i_{n_r+1} = n+1)$, where n_r is the number of customers in the route. We first need to determine the latest feasible time $\overline{t}_{i_j}^r$ to begin service at each customer (where $\overline{t}_{i_0}^r = \overline{t}_0^r$ and $\overline{t}_{i_{n_r+1}}^r = \overline{t}_{n+1}^r$ are the latest feasible departure and arrival times at the depot, respectively). A backward sweep of route r is applied from i_{n_r+1} to i_0 as follows:

$$\overline{t}_{i_{n_r+1}}^r \leftarrow b_{i_{n_r+1}},$$

$$\overline{t}_{i_j}^r \leftarrow \min\{\overline{t}_{i_{j+1}}^r - t_{i_j i_{j+1}} - s_{i_j}, b_{i_j}\}, \quad j = i_{n_r}, \ldots, i_0 = 0.$$

Once $\overline{t}^r_{i_0}$ has been obtained, a forward sweep is applied to reset the $\overline{t}^r_{i_j}$ values and get the latest feasible schedule:

$$\overline{t}^r_{i_j} \leftarrow \max\{\overline{t}^r_{i_{j-1}} + s_{i_{j-1}} + t_{i_{j-1}i_j}, a_{i_j}\},$$
$$j = i_1, \ldots, i_{n_r+1} = n+1.$$

The total route duration corresponds to $d^r_{\min} = \overline{t}^r_{n+1} - \overline{t}^r_0$, which is also the minimum duration, because the waiting time is minimized by serving the route at the latest feasible time.

3.3.2. Earliest departure and arrival times

We now want to determine the earliest departure and arrival times at the depot, $\underline{t}^r_{i_0} = \underline{t}^r_0$ and $\underline{t}^r_{i_{n_r+1}} = \underline{t}^r_{n+1}$, while maintaining the minimum route duration $d^r_{\min}$. Two cases should be considered:

Case 1. If there is no waiting time in the latest feasible schedule, then the latter can be "shifted" by δ^r time units, where

$$\delta^r = \min_{j=0,\ldots,n_r+1} (\overline{t}^r_{i_j} - a_{i_j}).$$

The time windows for arrival and departure at the depot are thus $[\underline{t}^r_0, \overline{t}^r_0]$ and $[\underline{t}^r_{n+1}, \overline{t}^r_{n+1}]$, where $\underline{t}^r_0 = \overline{t}^r_0 - \delta^r$ and $\underline{t}^r_{n+1} = \overline{t}^r_{n+1} - \delta^r$.

Case 2. If there is some waiting time in the latest feasible schedule, it is not possible to depart earlier from the depot without increasing the route duration. Consequently, we have $\underline{t}^r_0 = \overline{t}^r_0$ and $\underline{t}^r_{n+1} = \overline{t}^r_{n+1}$. In this case, the time windows for departure and arrival at the depot reduce to a single point. It is thus feasible to serve route r' after route r, and arc (r, r') is in the route graph, if $\underline{t}^r_{n+1} + \sigma^{r'} \leqslant \overline{t}^{r'}_0$ (assuming that the two routes are disjoint).

The original FDGG algorithm is used to find an elementary least cost path with time window constraints from the artificial start node to the artificial end node in the route graph. The departure time windows associated with each route are used for this purpose. That is, the vehicle must be back at the depot and ready to depart before the latest departure time of its next route. If the vehicle is ready before the earliest departure time, then it must wait. When arc (r, r') is added to the current path, route r' is included into the vehicle workday. In this case, the cost of route r' in Eq. (17) is incurred and the time consumed corresponds to the duration of route r', plus the setup time of route r', plus any waiting time before departure. The sequence of nodes in the least cost path obtained at the end corresponds to the sequence of routes in the vehicle workday. An example is provided below.

3.4. An example

Let us assume that we have $n = 5$ customer nodes indexed from 1 to 5 and a depot associated with nodes 0 and 6. We also assume that the distances and travel times are the same and correspond to the Euclidean metric. Table 1 indicates the coordinates and time windows for each node (taken from Solomon's test instances [13]).

Now, we want to calculate the departure and arrival time windows for route $r = 1$, $(0, 3, 1, 6)$, where $i_0 = 0$, $i_1 = 3$, $i_2 = 1$ and $i_3 = 6$. With travel times $t_{03} = 39.35$, $t_{31} = 3.00$ and $t_{16} = 38.07$ and a service time of 10 at customers 3 and 1, the backward sweep gives:

$$\overline{t}^1_6 = \infty,$$
$$\overline{t}^1_1 = \min\{\infty - 38.07 - 10,\ 874\} = 874,$$
$$\overline{t}^1_3 = \min\{874 - 3.00 - 10,\ 588\} = 588,$$
$$\overline{t}^1_0 = \min\{588 - 39.35,\ \infty\} = 548.65.$$

The forward sweep gives:

$$\overline{t}^1_0 = 548.65,$$
$$\overline{t}^1_3 = \max\{548.65 + 39.35,\ 473\} = 588,$$
$$\overline{t}^1_1 = \max\{588 + 3.00 + 10,\ 591\} = 601,$$
$$\overline{t}^1_6 = \max\{601 + 38.07 + 10,\ 0\} = 649.07.$$

Table 1
Customers with their time windows

Node	x	y	Time window
0	40	50	$[0, \infty]$
1	25	85	[591, 874]
2	22	75	[73, 350]
3	22	85	[473, 588]
4	20	80	[418, 913]
5	20	85	[40, 390]

Table 2
Time windows and cost of each route

Route	Departure	Arrival	Cost
0, 3, 1, 6	[538.65, 548.65]	[639.07, 649.07]	−82.82
0, 4, 3, 6	[421.57, 522.35]	[536.57, 637.35]	−82.46
0, 4, 1, 6	[537.88, 639.07]	[820.88, 922.07]	−82.05
0, 2, 5, 6	[42.20, 143.50]	[319.20, 420.50]	−81.94
0, 2, 6	[42.20, 113.80]	[319.20, 390.80]	−20.02
0, 4, 6	[381.95, 464.05]	[876.95, 959.05]	−9.52
0, 1, 6	[552.93, 639.07]	[35.93, 922.07]	−5.48
0, 3, 6	[433.65, 522.35]	[548.65, 637.35]	−2.92
0, 5, 6	[0.00, 90.62]	[349.69, 440.31]	−1.00

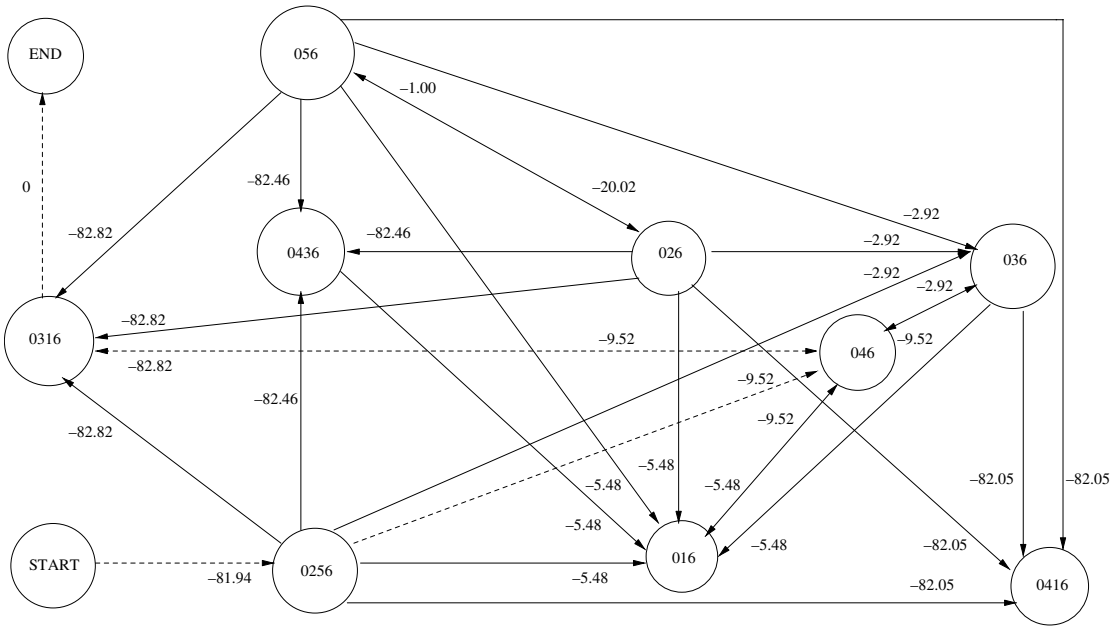

Fig. 1. Route graph.

Thus, the minimum route duration is equal to $649.07 - 548.65 = 100.42$. Since there is no waiting time in this schedule, we have

$$\delta^1 = \min\{548.65 - 0, 588 - 473, 601 - 591, 649.07 - 0\} = 10.$$

Consequently, the time windows for departure and arrival at the depot are $[\underline{t}_0^1, \overline{t}_0^1] = [538.65, 548.65]$ and $[\underline{t}_6^1, \overline{t}_6^1] = [639.07, 649.07]$, respectively. In Table 2, we show the time windows for all feasible routes generated in Phase 1, plus the costs obtained by setting α to $2\max_{i \in N}\{\max\{d_{0i},\ d_{i(n+1)}\}\} + 1$. With these, we can construct the route graph illustrated in Fig. 1 (where only one arc going out of the artificial start node and only one arc going into the artificial end node are shown, namely those that are part of the shortest path). The elementary shortest path obtained with FDGG is shown with a dotted line. The workday of the vehicle is thus made of route $(0, 2, 5, 6)$ followed by routes $(0, 4, 6)$ and $(0, 3, 1, 6)$.

4. Computational results

We have used the classical VRPTW instances of Solomon [13] for this study. Those are 100-customer Euclidean problems where distances and travel times are the same. There are six different classes of problems depending on the geographic location of customers (R: random; C: clustered; RC: mixed) and length of scheduling horizon (1: short horizon; 2: long horizon). In this study, problems of type 1 have been discarded because the short horizon does not allow the vehicle to serve many routes during its workday. We thus report results for problem classes R2 (11 instances), C2 (8 instances) and RC2 (8 instances) only. All tests were run on a 900 MHz Sun Ultra III with 8 GB of RAM.

Deadline constraints were added to Solomon's instances. Two different $t_{\max}$ values were tested for each problem class to evaluate how this parameter impact the algorithmic behavior. In the case of R2 and RC2, $t_{\max}$ was set to 75 and 90, while in the case of C2, it was set to 220 and 240. Note that $t_{\max}$ needs to be larger for class C2 because the service time at each customer is 30, as opposed to 10 for classes R2 and RC2. It should finally be noted that parameter β for the route loading time in Eq. (12) was set to 0.2 in all experiments.

4.1. Impact of dominance relation in phase 1

In this section, we first evaluate the impact of the modified dominance relation when feasible routes are generated with the elementary shortest path algorithm with resource constraints. Basically, we compare the number of routes produced with and without the dominance relation. In the latter case,

an exhaustive enumeration of all feasible routes is obtained. Table 3 shows the average number of routes obtained over all the experiments reported in Subsection 4.2 on problem classes R2, RC2 and C2, using $t_{max} = 75$, 75 and 220, respectively.

As we can see, by exploiting the dominance relation in the shortest path algorithm, the number of feasible routes generated in Phase 1 is reduced by approximately 20% (the routes that are eliminated are feasible, but are of no interest, as they are dominated by other routes). Clearly, this can only marginally impact the "combinatorial explosion" associated with increasing t_{max} values, but this is a nice way to exploit the shortest path algorithm already available to us. Note also that the number of feasible routes is much smaller for classes C2 and RC2, when compared with class R2. This is related to the presence of distant clusters in C2 and RC2 that prevent feasible routes to serve customers in different clusters.

Table 3
Number of routes generated in Phase 1

Problem class	Without dominance	With dominance	Reduction in percent (%)
R2	30695.8	21984.1	28.4
C2	2145.6	1732.3	19.3
RC2	4384.3	3450.5	21.3

4.2. Results on Solomon's instances

In Tables 4–9 the results obtained on Solomon's instances are reported. These results include smaller instances obtained by taking only the first 25 or 50 customers. In these tables, a particular instance is identified by its class, followed by its number (1–8,

Table 4
Results on class R2 with $t_{max} = 75$

Instance	# Routes Phase 1	# Routes Phase 2	Distance	# Cust. served	% Cust. served	# Cust. per route	CPU time (sec)
R201.25	122	7	535.77	16	64	2.28	0
R201.50	550	10	599.12	27	54	2.70	5
R201.100	4279	12	589.54	39	39	3.25	3579
R202.25	341	8	556.41	22	88	2.75	9
R202.50	2223	10	651.09	32	32	3.20	7478
R202.100	32,495						
R203.25	443	9	641.13	24	96	2.66	154
R203.50	3248						
R203.100	62,685						
R204.25	548	8	557.77	24	96	3.00	897
R204.50	4535						
R204.100	87,542						
R205.25	237	9	631.73	24	96	2.66	1
R205.50	1424	10	645.56	31	62	3.10	552
R205.100	16,399						
R206.25	425	9	633.42	25	100	2.77	92
R206.50	3059	10	629.29	34	68	3.40	47,968
R206.100	50,575						
R207.25	480	9	602.09	25	100	2.77	390
R207.50	3682						
R207.100	73,008						
R208.25	553	8	582.61	25	100	3.12	998
R208.50	4548						
R208.100	93,918						
R209.25	381	10	675.59	25	100	2.50	9
R209.50	2531	10	639.24	32	64	3.20	15,556
R209.100	42,731						
R210.25	397	9	635.48	24	96	2.66	58
R210.50	3160						
R210.100	51,537						
R211.25	597	8	588.01	25	100	3.12	196
R211.50	5067						
R211.100	124,008						

or 1–11, depending on the class) and number of customers. For example RC202.50 is the second instance of class RC2, where only the first 50 customers are considered. The other columns contain the number of feasible routes generated in Phase 1 (#Routes Phase 1), number of routes in the vehicle workday at the end of Phase 2 (#Routes Phase 2), total distance traveled by the vehicle, number of served customers, percentage of served customers, average number of customers per route and CPU time in seconds. When an entry is empty, the corresponding instance could not be solved. In some cases, we were able to generate feasible routes in Phase 1, but the associated route graph was too large to allow us to produce a solution.

We observed that the instances in class R2 are particularly hard to solve, due to a large number of feasible routes. In fact, only one solution was obtained on a 100-customer instance with $t_{max} = 75$ and none with $t_{max} = 90$. In the case of class RC2, 3 instances out of 8 with 100 customers were solved to optimality with $t_{max} = 75$, but only one with $t_{max} = 90$. In the case of class C2, 6 and 5 instances out of 8 were solved to optimality with $t_{max} = 220$ and 240, respectively. By increasing the value of t_{max}, solutions with fewer routes, more customers per route and a larger number of served customers are obtained. However, the number of feasible routes increases sharply, thus leading to substantial increases in computational requirements. Overall, the CPU times vary widely and range from only a few seconds to about one day, depending on problem size, problem class and t_{max} value.

It should be noted that, on the 100-customer instances, approximately 30% of the customers are served. This percentage increases to about 50–60%

Table 5
Results on class R2 with $t_{max} = 90$

Instance	# Routes Phase 1	# Routes Phase 2	Distance	# Cust. served	% Cust. served	# Cust. per route	CPU time (sec)
R201.25	217	7	511.32	20	80	2.85	1
R201.50	1263	10	623.33	28	56	2.80	37
R201.100	13,729						
R202.25	778	8	601.59	23	92	2.87	63
R202.50	6800						
R202.100							
R203.25	1097	8	598.1	24	96	3.00	855
R203.50	11,320						
R203.100							
R204.25	1435	6	506.55	24	96	4.00	4258
R204.50	7209						
R204.100							
R205.25	527	8	608.01	24	96	3.00	9
R205.50	4003	9	644.24	33	66	3.66	7749
R205.100							
R206.25	1034	8	582.32	25	100	3.12	723
R206.50	10,491						
R207.100							
R207.25	1236	7	543.53	25	100	3.57	2898
R207.50	13,539						
R207.100							
R208.25	1462	7	496.71	25	100	3.57	6488
R208.50	17,930						
R208.100							
R209.25	902	8	648.57	25	100	3.12	76
R209.50	8245						
R209.100	231,219						
R210.25	986	7	521.17	23	92	3.28	325
R210.50	11,144						
R210.100							
R211.25	1649	8	526.71	25	100	3.12	2015
R211.50	8123						
R211.100							

Table 6
Results on class RC2 with $t_{max} = 75$

Instance	# Routes Phase 1	# Routes Phase 2	Distance	# Cust. served	% Cust. served	# Cust. per route	CPU time (sec)
RC201.25	74	9	707.01	17	68	1.88	0
RC201.50	174	8	660.68	21	42	2.62	0
RC201.100	1562	12	620.35	31	31	2.58	139
RC202.25	167	8	654.75	20	80	2.50	1
RC202.50	395	8	689.55	25	50	3.12	15
RC202.100	5210						
RC203.25	252	8	673.2	23	92	2.87	9
RC203.50	587	8	668.27	27	54	3.37	708
RC203.100	9877						
RC204.25	319	8	673.93	24	96	3.00	40
RC204.50	865	8	653.95	29	58	3.62	7800
RC204.100	9141						
RC205.25	147	9	724.59	20	80	2.22	0
RC205.50	353	8	681.89	24	48	3.00	3
RC205.100	4037	11	591.77	34	34	3.09	13,511
RC206.25	138	7	610.81	21	84	3.00	0
RC206.50	334	8	664.86	25	50	3.12	3
RC206.100	4967	10	589.3	35	35	3.50	24,797
RC207.25	257	7	593.99	21	84	3.00	2
RC207.50	679	8	655.43	26	52	3.25	209
RC207.100	11,016						
RC208.25	384	8	678.09	24	96	3.00	56
RC208.50	995	8	658.28	28	56	3.50	3072
RC208.100	10,640						

Table 7
Results on class RC2 with $t_{max} = 90$

Instance	# Routes Phase 1	# Routes Phase 2	Distance	# Cust. served	% Cust. served	# Cust. per route	CPU time (sec)
RC201.25	132	7	659.15	18	72	2.57	0
RC201.50	312	7	610.17	22	44	3.14	0
RC201.100	4457	9	623.52	33	33	3.66	2389
RC202.25	323	6	540.1	20	80	3.33	4
RC202.50	911	7	626.36	28	56	4	112
RC202.100	20,000						
RC203.25	493	7	604.73	23	92	3.28	69
RG203.50	1412	7	623.56	31	62	4.42	4112
RC203.100	45,811						
RC204.25	708	7	645.19	25	100	3.57	282
RC204.50	2399	7	608.42	34	68	4.85	93,070
RC204.100	89,808						
RC205.25	264	7	588.13	20	80	2.85	1
RC205.50	753	7	644.81	26	52	3.71	23
RC205.100	15,161						
RC206.25	259	6	599.07	22	88	3.14	1
RC206.50	727	7	605.19	27	54	3.85	21
RC206.100	18,989						
RC207.25	560	6	535.31	22	88	3.66	14
RC207.50	1793	7	595.21	28	56	4	2114
RC207.100	53,233						
RC208.25	953	7	613.02	25	100	3.57	498
RG208.50	3110	7	617.92	31	62	4.42	47,057
RC208.100	115,098						

Table 8
Results on class C2 with $t_{max} = 220$

Instance	# Routes Phase 1	# Routes Phase 2	Distance	# Cust. served	% Cust. served	# Cust. per route	CPU time (sec)
C201.25	102	10	573.48	22	88	2.20	0
C201.50	340	11	581.56	28	56	2.54	0
C201.100	1204	12	548.1	31	31	2.58	23
C202.25	287	10	616.49	24	96	2.40	4
C202.50	1024	11	573.21	30	60	2.72	539
C202.100	3557	11	500.5	31	31	2.81	33,375
C203.25	410	10	647.93	25	100	2.50	133
C203.50	1668	11	596.25	31	62	2.81	66,527
C203.100	5701						
C204.25	525	10	628.22	25	100	2.50	657
C204.50	2129						
C204.100	7319						
C205.25	140	10	584.47	24	96	2.40	0
C205.50	548	12	603.03	30	60	2.50	3
C205.100	2047	11	502.38	31	31	2.81	221
C206.25	166	11	648.92	25	100	2.27	0
C206.50	718	11	569.9	30	60	2.72	17
C206.100	2641	11	507.76	32	32	2.90	920
C207.25	282	10	607.29	25	100	2.50	6
C207.50	1066	11	591.45	31	62	2.81	360
C207.100	3078	11	495.47	32	32	2.90	3294
C208.25	210	10	613.2	25	100	2.50	1
C208.50	882	11	566.98	31	62	2.81	54
C208.100	3244	11	488.67	32	32	2.90	3002

Table 9
Results on class C2 with $t_{max} = 240$

Instance	# Routes Phase 1	# Routes Phase 2	Distance	# Cust. served	% Cust. served	# Cust. per route	CPU time (sec)
C201.25	139	9	515.67	23	92	2.55	0
C201.50	505	10	582.14	29	58	2.90	2
C201.100	1804	11	538.84	31	31	2.81	67
C202.25	634	9	574.14	25	100	2.77	21
C202.50	2461	11	548.39	30	60	2.72	4606
C202.100	9064						
C203.25	1288	9	533.87	25	100	2.77	1159
C203.50	5171						
C203.100							
C204.25	1661	9	535.22	25	100	2.77	5687
C204.50	7405						
C204.100							
C205.25	269	9	538.87	25	100	2.77	0
C205.50	965	10	558.24	30	60	3.00	19
C205.100	3575	11	502.15	31	31	2.81	1025
C206.25	334	9	536.09	25	100	2.77	1
C206.50	1428	11	582.58	31	62	2.81	107
C206.100	5182	11	507.76	32	32	2.90	7774
C207.25	639	9	529.77	25	100	2.77	30
C207.50	2518	11	561.71	31	62	2.81	3800
C207.100	6421	11	495.47	32	32	2.90	21,084
C208.25	416	9	527.84	25	100	2.77	2
C208.50	1847	11	553.63	31	62	2.81	370
C208.100	6881	11	488.67	32	32	2.90	26,461

Table 10
Results on class RC2 for individual routes

Instance	# Routes Phase 1	# Routes Phase 2	Distance	# Cust. served	% Cust. served	# Cust. per route	CPU time (sec)
rc201.21	241	6	475.25	19	90	3.16	0
rc201.27	384	7	626.14	22	82	3.14	0
rc201.26	225	8	728.52	23	88	2.87	1
rc201.26	464	6	468.81	22	84	3.66	1
rc202.38	2296	8	661.3	28	73	3.50	3151
rc202.30	873	7	591.42	25	83	3.57	43
rc202.32	1738	9	633.83	28	87	3.11	790
rc203.27	1908	7	522.76	25	92	3.57	831
rc203.29	4618	9	667.31	28	96	3.11	43,729
rc205.29	1269	8	541.54	26	89	3.25	118
rc205.22	348	7	646.45	20	90	2.85	1
rc205.28	751	8	680.06	24	85	3.00	22
rc205.21	886	6	444.7	20	95	3.33	12
rc206.32	2409	8	604.25	27	84	3.37	359
rc206.35	1820	8	652.39	27	77	3.37	297
rc206.33	1821	8	619.66	29	87	3.62	217
rc207.27	1869	7	590.36	26	96	3.71	939
rc207.35	5519	8	572.44	30	85	3.75	44,356
rc207.38	2610	7	656.41	28	73	4.00	2441

on the 50-customer instances and to 70–100% on the smallest instances with 25 customers. The small percentages observed in the case of the 100-customer instances is not a surprise, given that the latter have been designed for multiple vehicles. That is, when we consider a relatively short time period within the scheduling horizon and gather all customers with a time window that falls into that period, there are clearly too many for a single vehicle to serve them all (while this could be possible with many vehicles).

4.3. Results on individual routes

In this subsection, we consider instances created from individual routes in the best known solutions for class RC2 (as reported in [1,4,12]). That is, our algorithm was applied only to the subset of customers associated with a particular route. As a feasible route in a VRPTW solution is made of customers with time windows that are well distributed over the time horizon, we expect our algorithm to include a large fraction of these customers in the vehicle workday. Overall, the new created instances contain between 21 and 38 customers.

Results obtained with $t_{\max} = 90$ are shown in Table 10. In this table, an identifier like RC201.21 is the instance created with the 21 customers taken from one of the routes in the best know solution for RC201. As we can see, the number of customers per route has increased, when compared with the 25-customer instances in Table 7. Also, the percentages of served customers on instances with 30 customers or less are better than those of the 25-customer instances in Table 7. In fact, we could not expect much better, given the additional deadline constraint that forces the vehicle to travel back to the depot and reload, thus leading to a waste of time.

5. Conclusion

This paper has described an exact algorithm for solving a routing problem where a vehicle performs several routes over the scheduling horizon. The results indicate that this algorithm is very sensitive to the deadline constraint. When this constraint is not tight enough, the number of feasible routes "explodes" and becomes too large to allow the algorithm to produce a solution. Future developments will now be aimed at considering problems that are closer to real-world applications. First, a fleet of many vehicles will be considered. Then, a dynamic version of the problem will be tackled, where new incoming customer requests have to be integrated in real time into the current solution.

Acknowledgements

Financial support for this work was provided by the Canadian Natural Sciences and Engineering Research Council (NSERC) and by the Quebec Fonds

pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR). This support is gratefully acknowledged.

References

[1] R. Bent, P. van Hentenryck, A two-stage hybrid local search for the vehicle routing problem with time windows, Transportation Science 38 (2004) 515–530.
[2] A.M. Campbell, M. Savelsbergh, Decision support for consumer direct grocery initiatives, Transportation Science 39 (2005) 313–327.
[3] A.M. Campbell, M. Savelsbergh, Efficient insertion heuristics for vehicle routing and scheduling problems, Transportation Science 38 (2004) 369–378.
[4] Z.J. Czech, P. Czarnas, A parallel simulated annealing for the vehicle routing problem with time windows, in: Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-Based Processing, Canary Islands, Spain, 2002, pp. 376–383.
[5] J. Desrosiers, Y. Dumas, M.M. Solomon, F. Soumis, Time constrained routing and scheduling, in: M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser (Eds.), Network Routing, North Holland, 1995, pp. 35–139.
[6] D. Feillet, P. Dejax, M. Gendreau, C. Gueguen, An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems, Networks 44 (2004) 216–229.
[7] B. Fleischmann, The vehicle routing problem with multiple use of vehicles, Working Paper, Fachbereich Wirtschaftswissenschaften, Universität Hamburg, Germany, 1990.
[8] H.K. Kilpala, The impact of electronic commerce on transport & logistics in the retail grocery industry, M.Sc. Thesis, University of Oulu, Finland, 1999.
[9] I.L. Lin, H.S. Mahmassani, Can online grocers deliver? Some logistics considerations, Transportation Research Record 1817 (2002) 17–24.
[10] M. Punakivi, J. Saranen, Identifying the success factors in e-grocery home delivery, International Journal of Retail and Distribution Management 29 (2001) 156–163.
[11] M. Punakivi, Comparing alternative home delivery models for e-grocery business, Doctoral Dissertation, Department of Industrial Engineering and Management, Helsinki University of Technology, Finland, 2003.
[12] L.-M. Rousseau, M. Gendreau, G. Pesant, Using constraint-based operators to solve the vehicle routing problem with time windows, Journal of Heuristics 8 (2002) 43–58.
[13] M.M. Solomon, Algorithms for the vehicle routing and scheduling problem with time window constraints, Operations Research 35 (1987) 254–265.
[14] É.D. Taillard, G. Laporte, M. Gendreau, Vehicle routing with multiple use of vehicles, Journal of the Operational Research Society 47 (1996) 1065–1070.
[15] H. Yrjölä, Physical distribution considerations for electronic grocery shopping, International Journal of Physical Distribution and Logistics Management 31 (2001) 746–761.