Solving the Vehicle Routing Problem with Lunch Break Arising in the Furniture Delivery Industry

Leandro C. Coelho, Jean-Philippe Gagliardi, Jacques Renaud, Angel Ruiz

CIRRELT and Faculté des sciences de l'administration, Université Laval, Québec, Canada

{leandro.coelho, jean-philippe.gagliardi, jacques.renaud, angel.ruiz}@cirrelt.ca

December 1, 2015

Abstract

In this paper we solve the Vehicle Routing Problem with Lunch Break (VRPLB) which arises when drivers must take pauses during their shift, for example, for lunch breaks. Driver breaks have already been considered in long haul transportation when drivers must rest during their travel, but the underlying optimization problem remains difficult and few contributions can be found for less than truckload and last mile distribution contexts. This problem, which appears in the furniture delivery industry, includes rich features such as time windows and heterogeneous vehicles. In this paper we evaluate the performance of a new mathematical formulation for the VRPLB and of a fast and high performing heuristic. The mixed integer linear programming formulation has the disadvantage of roughly doubling the number of nodes, and thus significantly increasing the size of the distance matrix and the number of variables. Consequently, standard branch-and-bound algorithms are only capable of solving small-sized instances. In order to tackle large instances provided by an industrial partner, we propose a fast multi-start randomized local search heuristic tailored for the

VRPLB, which is shown to be very efficient. Through a series of computational experiments, we show that solving the VRPLB without explicitly considering the pauses during the optimization process can lead to a number of infeasibilities. These results demonstrate the importance of integrating drivers pauses in the resolution process.

Keywords: vehicle routing problem with lunch break; time windows; heterogeneous fleet; pauses; exact formulation; multi-start local search.

1 Introduction

In this paper we introduce, model and solve a rich vehicle routing problem with lunch break (VRPLB) proposed by an industrial partner dealing with furniture delivery. This pause scheduling problem arises, e.g., in retail delivery, parcel and mail delivery, waste collection, and home blood sampling collection [5, 17, 24]. In the VRPLB, drivers must make a pause during their shift to respect the law, union rules, and/or company regulations. The VRPLB is a generalization of several classes of the well-known VRP [18], notably the VRP with time windows in which each node must be visited within a certain time interval. In our context, **not only customers have time windows, but for each vehicle route a** pause must also be taken within an interval**. Since** its location is unknown a priori one cannot simply add pause nodes to the problem, as the distances between customers must remain unchanged. Moreover, one does not know a priori how many pauses will be needed, as the numbrer of routes is typically a decision variable.

Pauses are often seen in long-haul transportation when drivers are required to stop and rest after a given number of driving hours to avoid excessively long working shifts [4, 11, 12, 13, 14, 15, 31]. In last mile distribution, pauses are also present, for example, when drivers have time allotted for lunch breaks. **Also, when many customers must be visited, the timing of the pauses highly affects the feasibility of the solution with respect to their time windows, whereas in long haul transportation, with only one customer to be visited by a full truckload delivery, the position of the pause does not impact the arrival time.**

Although very practical, the literature on VRPLB is rather scarce, and we are aware of only two papers modeling it exactly [7, 25]. Buhrkal et al. [7] add binary variables indicating whether the pause was taken between a visit of two consecutive nodes in the trip, whereas Sahoo et al. [25] model it by adding

a single lunch break to the problem by means of a single dummy node which is then linked to all other location nodes. Nevertheless, neither Buhrkal et al. [7] nor Sahoo et al. [25] used their models in the optimization phase, relying on heuristics, so the performance of these models is still unknown. Finally, Vidal et al. [32] work in the context of a unified framework based on genetic operators to solve the VRPLB and other rich VRP problems heuristically without proposing a mathematical model.

Regarding industrial applications of the VRP, only a few works report results based on real cases and even fewer report industrial implementations. Among them, the food and soft-drink industries are the most prominent [16, 21, 27, 28]. Other applications arise in different industries, including lubricating oil distribution [23, 29], waste collection [5, 17], industrial gases [9, 10], petroleum products [3, 6], and cash [30]. However, even if these articles deal with real applications, drivers pauses were not considered in an exact and extensive fashion.

Our study is motivated by the request of an industrial partner facing this problem. Thus, we model, solve exactly and evaluate drivers pauses as a real-world constraint, and we compare the solutions of an exact algorithm with those of our fast heuristic. Despite a large body of research on VRPs, many industrial applications remain open and in some cases can only be handled with heuristics, due to the lack of a mathematical formulation and algorithms capable of handling them. One of these constraints is related to drivers pauses. Notwithstanding, thousands of miles of drivers routes are planned every day. In practice, routes are often planned disregarding the pause, which is later inserted into the final solution, yielding a poor approximation. This strategy can be applied for problems in which the pause separates two delivery periods, similar to a multi-period problem [1]. However, in a short planning horizon such as daily delivery in which our problem is defined, the pause does not lead to a multi-period problem. In fact, this strategy does not suit the retail industry where many customer deliveries should be done within some (tight) time windows and where service quality is highly important. In that context the timing of the pause becomes crucial.

The main contributions of this paper lie in formulating the problem as a mixed-integer linear programming model, considering time windows and driver pauses without the need for specific pause variables as in Buhrkal et al. [7] and Sahoo et al. [25]; in developing a fast and efficient multi-start randomized local search heuristic capable of handling this and other practical constraints; and finally in demonstrating through computational experiments the potential drawbacks of solving distribution problems with time windows without explicitly considering drivers pauses.

The remainder of this paper is organized as follows. In Section 2 we formally describe the VRPLB. In Section 3 we propose a mixed-integer linear programming formulation to model the VRPLB. The

description of our multi-start heuristic is presented in Section 4, which is followed by the computational experiments in Section 5. Conclusions are presented in Section 6.

2 Description of the problem

The VRPLB is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V} = \{0, \ldots, n+1\}$ is the vertex set and $\mathcal{A}$ is the arc set. Vertices 0 and $n+1$ correspond to two copies of the depot, while the remaining vertices of $\mathcal{V}' = \{1, \ldots, n\}$ represent the customers requesting a delivery. Each arc $(i, j) \in \mathcal{A}$ is associated with a travel distance d_{ij} and a travel time l_{ij}. Each customer i is associated with a service time S_i, a delivery weight w_i, and a delivery volume v_i. The service of a customer i must start within a time window $[a_i, b_i]$. We assume that all time windows are feasible, i.e., all customers can be reached from the depot within their time windows.

A heterogeneous fleet of K vehicles is located at the depot. Each vehicle k is associated with a volume capacity V_k and a weight capacity W_k. Vehicles routes must begin between $[a_0, b_0]$ and must end before b_{n+1} and, within these limits, each route must respect a maximum working time L_k. Vehicles are allowed to wait between two customers, and a lunch period lasting S_p units of time must be scheduled to start between $[a_p, b_p]$ in each route. We assume pauses are made at a customer location, as is the case of our industrial partner. In this context, the goal is to minimize the total length of the routes in terms of traveling distance.

3 Mathematical formulation

Since drivers' pauses can be taken at a customer site right after the service has been completed, or at a customer site just before starting the service, distances from the pauses' sites to the customers locations need to consider both cases. **Without loss of generality, in our model we assume that pauses are taken immediately after visiting a customer. However, the actual pause can be taken anywhere along the route, as long as it happens before the next visit, and the solution will remain the same.** We provide in Sections 3.1 and 3.2 a detailed description of the transformations required to account for a pause in each route, and in Section 3.3 a mathematical formulation for the VRPLB.

3.1 Extended distance matrix

Consider an instance with n customers, and its distance matrix d_{ij} defined over $\mathcal{A}$. The distance matrix d_{ij} contains $n+2$ rows and $n+2$ columns. In order to account for pause and time windows, we will work with an extended distance matrix d'_{ij} containing $2n+4$ rows and columns. For notation purposes, we define a new set $\mathcal{W}$ containing all nodes of $\mathcal{V} \cup \{n+2, \ldots, 2n+3\}$. In the extended distance matrix d'_{ij}, indices $n+2$ to $2n+2$ refer to pause nodes, one for each original node, and node $2n+3$ indicates the depot.

$$
d'_{ij} = \left[\begin{array}{cccc|ccc|c}
\multicolumn{4}{c|}{\multirow{3}{*}{A_1}} & \multicolumn{3}{c|}{\multirow{3}{*}{A_2}} & \multirow{3}{*}{A_5} \\
 & & & & & & & \\
 & & & & & & & \\
\hline
\multicolumn{4}{c|}{\multirow{3}{*}{A_3}} & \multicolumn{3}{c|}{\multirow{3}{*}{A_4}} & \multirow{3}{*}{A_6} \\
 & & & & & & & \\
 & & & & & & & \\
\hline
\multicolumn{8}{c}{A_7}
\end{array}\right]
$$

Obviously, the upper-left part of matrix d'_{ij}, identified as A_1, corresponds to d_{ij}:

$$d'_{ij} = d_{ij} \qquad i, j \in \mathcal{V}. \tag{1}$$

In order to model the problem, one needs to represent each location as two different nodes to adequately account for pauses. The reason is that there is no cost related to traveling to a pause location, but we must keep track of the last node visited before the pause in order to compute the appropriate distance to the next node after the pause. Thus, node $n+2$ models a pause made immediately after leaving the depot (node 0); node $n+3$ models a pause made immediately after visiting node 1, until node $2n+2$ which represents a pause taken immediately after visiting node n. Since there is no cost for traveling to the pause, distances in the submatrix A_2 of d'_{ij} are modeled as follows:

$$d_{ij}^{'} = \begin{cases} 0 & \text{if } j = i + n + 2 \\ \infty & \text{otherwise} \end{cases} \qquad i \in \mathcal{V} \quad j \in \mathcal{W} \backslash (\mathcal{V} \cup \{2n + 3\}) \,. \tag{2}$$

Then, one needs to model the distance from each pause node back to each customer $i \in \mathcal{V}$. If the vehicle visits the pause node associated with customer i, it cannot go back to i, thus the associated distance will be infinity. However, the vehicle can travel to any other customer j with the same original distance d_{ij}. Thus, the elements of $d_{ij}^{'}$ in the submatrix A_3 are modeled as:

$$d_{ij}^{'} = \begin{cases} \infty & \text{if } j = i - n - 2 \\ d_{i-n-2,j} & \text{otherwise} \end{cases} \qquad i \in \mathcal{W} \backslash (\mathcal{V} \cup \{2n + 3\}) \quad j \in \mathcal{V}. \tag{3}$$

Submatrix A_4 in $d_{ij}^{'}$ indicates an arc linking two pause nodes, which is forbidden. Thus, all its elements are equal to infinity:

$$d_{ij}^{'} = \infty \qquad i, j \in \mathcal{W} \backslash (\mathcal{V} \cup \{2n + 3\}) \,. \tag{4}$$

Finally, one needs to model the last row and last column of the extended distance matrix, which refers to trips from and to the arrival depot node. Submatrix A_5 refers to trips from customers $i \in \mathcal{V}$ to the depot node $2n + 3$, which are all equal to d_{i0}:

$$d_{i,2n+3}^{'} = d_{i0} = d_{i,n+1} \qquad i \in \mathcal{V}. \tag{5}$$

Submatrix A_6 is related to trips from pause nodes to the arrival depot, which have the same distance as the trips from the original nodes associated with each pause node:

$$d_{i+n+2,2n+3}^{'} = d_{i0} \qquad i \in \mathcal{V}. \tag{6}$$

The last row of the extended distance matrix $d_{ij}^{'}$ refers to trips from the arrival depot node to all other nodes. Since the arrival depot node is a sink, no vehicle is allowed to leave it, and all the elements in submatrix A_7 are equal to infinity:

$$d_{2n+3,j}^{'} = \infty \qquad i \in \mathcal{W}. \tag{7}$$

Traveling times need to be updated in a similar fashion as the distance matrix.

3.2 Updating time windows and service duration parameters

The creation of the extended distance matrix leads to larger time window and service duration matrices as well. These updates must take into account the node they refer to (customers, pauses, or depot).

Using the nomenclature defined in the previous section, we now provide the time windows and service duration information for all nodes in $\mathcal{W}$. Nodes 0 to $n+1$ refer to the original **outgoing** depot, customers**, and incoming depot** of the problem, and thus their time windows and service duration remain unchanged.

All pause nodes are identified by indices $n+2$ to $2n+2$, and their time windows are all equal to $[a_p, b_p]$. The service duration when these nodes are visited is always equal to S_p.

The last node $2n+3$ indicates the arrival depot, thus its service time is equal to zero. Each vehicle must return to the depot by b_{n+1}, so the time window of the returning depot is $[a_0, b_{n+1}]$.

3.3 Mixed-integer linear programming formulation

Using the extended distance matrix, we define an extended arc set $\mathcal{Z}$, with arcs $(i,j) \in \mathcal{Z}$, $i,j \in \mathcal{W}$. We formulate the VRPLB using the following variables. Binary routing variables x_{ij}^k are equal to one if and only if arc $(i,j) \in \mathcal{Z}$, is used on the route of vehicle $k \in \mathcal{K}$. Binary variables y_i^k are equal to one if and only if node $i \in \mathcal{W}$ is visited by vehicle k. Continuous variables s_i^k represent the starting time of the service for customer i by vehicle k. Let M be a big number. The problem is then formulated as follows:

$$\text{minimize} \sum_{(i,j)\in\mathcal{Z}} \sum_{k\in\mathcal{K}} d_{ij} x_{ij}^k \tag{8}$$

subject to the following constraints:

$$\sum_{i\in\mathcal{V}'} v_i y_i^k \leq V_k y_0^k \qquad k \in \mathcal{K} \tag{9}$$

$$\sum_{i\in\mathcal{V}'} w_i y_i^k \leq W_k y_0^k \qquad k \in \mathcal{K} \tag{10}$$

$$\sum_{k\in\mathcal{K}} y_i^k = 1 \qquad i \in \mathcal{V}' \tag{11}$$

$$\sum_{i\in\mathcal{W}\setminus(\mathcal{V}\cup\{2n+3\})} y_i^k \leq 1 \qquad k \in \mathcal{K} \tag{12}$$

$$\sum_{i\in\mathcal{W}\setminus(\mathcal{V}\cup\{2n+3\})} y_i^k = y_0^k \qquad k \in \mathcal{K} \tag{13}$$

$$n y_0^k \geq \sum_{i\in\mathcal{V}'} y_i^k \qquad k \in \mathcal{K} \tag{14}$$

$$\sum_{i\in\mathcal{W}\setminus\{0,2n+3\}} x_{0i}^k = y_0^k \qquad k \in \mathcal{K} \tag{15}$$

$$y_{2n+3}^k = y_0^k \qquad k \in \mathcal{K} \tag{16}$$

$$\sum_{i\in\mathcal{W}\setminus\{0,2n+3\}} x_{i,2n+3}^k = y_{2n+3}^k \qquad k \in \mathcal{K} \tag{17}$$

$$\sum_{j\in\mathcal{W}} x_{ij}^k + \sum_{j\in\mathcal{W}} x_{ji}^k = 2y_i^k \quad i \in \mathcal{W}\setminus\{0, 2n+3\} \quad k \in \mathcal{K} \tag{18}$$

$$\sum_{i\in\mathcal{W}} x_{ij}^k \leq 1 \qquad i \in \mathcal{W} \quad k \in \mathcal{K} \tag{19}$$

$$\sum_{i\in\mathcal{W}} x_{ji}^k \leq 1 \qquad i \in \mathcal{W} \quad k \in \mathcal{K} \tag{20}$$

$$s_i^k + S_i + l_{ij} - M\left(1 - x_{ij}^k\right) \leq s_j^k \qquad i, j \in \mathcal{W} \quad k \in \mathcal{K} \tag{21}$$

$$s_i^k \geq a_i \qquad i \in \mathcal{W} \quad k \in \mathcal{K} \tag{22}$$

$$s_i^k \leq b_i \qquad i \in \mathcal{W} \quad k \in \mathcal{K} \tag{23}$$

$$s_{2n+3}^k - s_0^k \leq L_k \qquad k \in \mathcal{K} \tag{24}$$

$$x_{ij}^k, y_i^k \in \{0,1\} \qquad (i,j) \in \mathcal{Z} \quad i \in \mathcal{W} \quad k \in \mathcal{K}. \tag{25}$$

The objective function (8) aims at minimizing the total vehicle routing costs/distance. Constraints (9) and (10) impose vehicle capacities with respect to the total volume and weight, respectively. Constraints (11) impose that all customers must be visited by exactly one vehicle. Constraints (12) impose that at most one pause node is visited by each vehicle, while constraints (13) require that a pause node is visited if the vehicle is used. Constraints (14) impose that the vehicle visits the depot if any customer is assigned to it, and constraints (15) ensure that the vehicle leaves the depot. Constraints (16) ensure that if a vehicle leaves the depot node 0 it should return to the depot node

$2n + 3$. Constraints (17) ensure that if the vehicle should return to the depot, one arc towards it is used. Constraints (18) are degree constraints, and constraints (19) and (20) ensure that there are at most one incoming and one outgoing arc for each node. Time windows and subtour elimination are imposed by means of constraints (21). Bounds on the time windows for the beginning of the service on every node are imposed through constraints (22) and (23). Shift duration constraints are imposed through constraints (24). Integrality conditions are imposed by constraints (25).

4 Multi-start randomized local search

We will later show that the formulation presented in Section 3.3 is too difficult to be solved even for small sized instances of the VRPLB. Thus, we propose an insertion and improvement heuristics embedded into a multi-start randomized local search, which will help us demonstrate the impact of managing driver pauses in large artificial and real-life instances.

The multi-start algorithm contains four main procedures, which are repeated a predetermined number of times (20 in our implementation). We describe the initialization phase in Section 4.1. An important procedure in our algorithm is related to the way we open new routes. We describe in Section 4.2 the procedure which opens one or several new routes as required. The randomized procedure to route customers is presented in Section 4.3. A local improvement procedure is described in Section 4.4, and a route compression phase is presented in Section 4.5.

In our algorithm, pauses are handled by adding a dummy customer with a time window $[a_p, b_p]$ and a service time S_p to each route.

It is worth mentioning that because of the practical nature of the problem, and due to the limited number of vehicles and to tight customer time windows, it may not be possible to include all customers in a solution. These unserved customers are referred to as *exceptions*. In practice, the list of exceptions is handled by the transportation department having the ability to violate some constraints in order to fit each exception or the customer service department who will contact the associated customer in order to schedule another delivery date. Such a set of customers not to be visited has already been treated in the literature [2, 8].

4.1 Initialization

The first phase of the algorithm creates and initializes the different data sets that will be used during the routing process. Let $\mathcal{R}$ be the set of active routes and $\mathcal{E}$ be the set of exceptions, both initially empty. Let $\mathcal{C}$ be a set containing all customers. After these initializations, the following procedure is executed in order to open one or several routes and to sequentially assign customers in $\mathcal{C}$ until all customers have been assigned to routes or to the set $\mathcal{E}$.

4.2 Opening new routes

One important feature of our algorithm is that, unlike other classic approaches which open one new route at a time, it can open up to π new routes simultaneously whenever the existing routes cannot accommodate the remaining customers in $\mathcal{C}$. The motivation for opening several routes instead of only one is to avoid having very busy routes which are concentrated in a specific area, which leads to very high costs when customers away from this cluster need to be visited. Thus, if two or more routes are open, the procedure tries to initialize them in different directions aiming at a better geographic coverage.

The number of routes to open depends on the number of the remaining customers to be visited, and on their needs. In particular, the procedure computes an estimate on the number of routes required to fulfill the deliveries to the remaining customers. This estimation is based on three ratios regarding the capacity of the vehicles in terms of both volume and weight, and the traveling and service times. The first two ratios are computed as the total required volume (weight) of the deliveries divided by the average capacity of the vehicles. The third one is based on the maximum shift duration, and is equal to the ratio between the estimation of the total time required to visit all the remaining customers and the maximum shift duration. The total time required to visit all customers is estimated as the average traveling time between each unvisited customer i and all the nodes in $\mathcal{C} \cup \{0\}$ plus their required service times. The maximum of these three ratios provides the number of routes to be opened by this procedure, bounded by the number of available vehicles and by a parameter indicating the maximum number of routes to be opened per iteration. New routes are opened using the vehicles with larger capacity.

Each of the new routes is initialized with a pause and by adding a customer to it in the following manner. If only one route is created, we include in this route the customer in $\mathcal{C}$ the farthest customer from the depot. If two routes are opened, two customers i and j in $\mathcal{C}$ are selected in such a way

that the total distance $d_{0i} + d_{ij} + d_{j0}$ is maximized. If three routes are created, three customers i, j and k are selected such that $d_{0i} + d_{0j} + d_{0k} + d_{ij} + d_{jk} + d_{ki}$ is maximized. We have limited π to three as per our preliminary numerical experiments. This way, new routes will tend to cover opposite regions. This approach was mainly motivated by the fact that the territory of the province of Quebec is very large and sparsely populated. According to our experiments, this issues would be less critical in densely populated areas.

Each of the new routes are added to $\mathcal{R}$, and the customers that have been assigned to them are removed from $\mathcal{C}$.

4.3 Routing customers

This step inserts customers from $\mathcal{C}$ into existing routes $\mathcal{R}$. At the end of this step, all customers will have been routed or set as exceptions. For each customer, we use the following procedure to find the best route and to place the customer in the best position.

We try inserting each customer in each possible position of all the existing routes. The insertion between two customers is feasible only when it respects capacity constraints, all time windows, total route duration, and end of route time. When all the routes and all the insertion positions have been tested, the procedure returns for each customer the best insertion position and cost (see line 7 of Algorithm 1). The next customer to be inserted is obtained by dividing the insertion cost by a random value following a uniform distribution taken from the interval $[\alpha, 1]$ (see lines 8–11 of Algorithm 1), where $\alpha = 1 - \frac{\text{iteration}-1}{\text{max iterations}}$, and selecting the customer with the smallest value. This means that the first iteration is fully deterministic, while the remaining successive iterations are each more randomized than the previous one.

If an insertion is possible, we remove the selected customer from $\mathcal{C}$, update $\mathcal{R}$, and repeat the procedure. Otherwise, we try to open a new route using the procedure described in Section 4.2. If no more vehicles are available, the remaining customers are added to the exception set $\mathcal{E}$. We provide a sketch of this procedure in Algorithm 1.

4.4 Local improvement

The local improvement phase consists of two neighborhood search heuristics which are applied to all routes in $\mathcal{R}$ in order to decrease their total distance. We first apply an intra-route search, followed by

Algorithm 1 Routing customers (Section 4.3)

1: best_client = 0
2: best_ratio = ∞
3: **if** $\mathcal{C} = \emptyset$ **then**
4: Return routes and $\mathcal{E}$.
5: **end if**
6: **for** $i \in \mathcal{C}$ **do**
7: $c_i \leftarrow$ best feasible insertion for customer i
8: $c_i^{'} \leftarrow \frac{c_i}{rand[\alpha,1]}$
9: **if** $c_i^{'} <$ best_ratio **then**
10: best_ratio $\leftarrow c_i^{'}$
11: best_client $\leftarrow i$
12: **end if**
13: **end for**
14: **if** best_client $\neq 0$ **then**
15: Insert best_client in its best position
16: $\mathcal{C} \leftarrow \mathcal{C} \backslash \{\text{best\_client}\}$
17: Return call `routing_customers()`.
18: **else**
19: **if** there are vehicles available **then**
20: Call `open_new_route(s)` procedure (see Section 4.2)
21: Return call `routing_customers()`.
22: **else**
23: Insert all customers from $\mathcal{C}$ in $\mathcal{E}$
24: **end if**
25: **end if**
26: Return routes and $\mathcal{E}$.

an inter-routes search.

Two feasibility checks are performed in order to accept a move. First, we evaluate the vehicle capacity, both in terms of volume and weight, as well as the total route duration. The second feasibility check concerns the time windows of all the customers in the route.

The intra-route improvement considers one route at a time. It applies the 3-opt algorithm of Lin [19] in which we also check for time window feasibility. Within each iteration, the first improving move is applied and the complete 3-opt algorithm is applied until no more improvements can be obtained, with a worst case complexity of $O(n^4)$ for each iteration.

Then, the inter-routes improvement procedure considers all routes in $\mathcal{R} = \{r_1, \ldots, r_K\}$ and tries to improve each pair of routes r_i ($i = 1, \ldots, K-1$) and r_j ($j = i+1, \ldots, K$) by exchanging customers between them. As proposed by Renaud et al. [22] for the fleet size and mixed vehicle routing problem, 11 exchanges are evaluated. These exchanges are detailed next and are applied to all possible chains of four consecutive vertices between routes r_i and r_j. The depot and the dummy nodes representing the drivers' pauses are never exchanged. The first improving move is applied and the procedure continues as long as improvements are obtained. Once all the possible pairs of routes have been considered, the procedure restarts by evaluating only the pairs of routes for which at least one of the routes have been modified during the previous iteration. This procedure can be viewed as a restriction of the 2-interchange procedure of Osman [20], however we concentrate on more promising moves yielding a much faster heuristic. The worst case complexity of these moves is $O(n^3)$ for each iteration.

The local search procedures are based on the 11 operations described by Renaud et al. [22]. A brief explanation follows. Let (i_1, i_2, i_3, i_4) and (j_1, j_2, j_3, j_4) be two chains of four vertices from routes r_i and $r_j \in \mathcal{R}$. The following 11 moves are considered are: (1) place i_2 between j_1 and j_2; (2) place j_2 between i_1 and i_2; (3) swap i_2 and j_2; (4) move i_2 and i_3 to r_j, inserting (i_2, i_3) between j_1 and j_2; (5) move i_2 and i_3 to r_j, inserting (i_3, i_2) between j_1 and j_2; (6) move j_2 and j_3 to r_i, inserting (j_2, j_3) between i_1 and i_2; (7) move j_2 and j_3 to r_i, inserting (j_3, j_2) between i_1 and i_2; (8) swap i_2 and j_2, swap i_3 and j_3; (9) swap i_2 and j_3, swap i_3 and j_2; (10) replace i_2 and i_3 by j_2 and j_3, respectively, and replace j_2 and j_3 by i_3 and i_2, respectively; (11) replace i_2 and i_3 by j_3 and j_2, respectively, and replace j_2 and j_3 by i_2 and i_3, respectively. Moves 1 to 3 correspond to the 1-interchange procedure of Osman [20], and moves 4 to 11 represent a subset of the 2-interchange exchanges tested by Osman [20].

4.5 Route compression

This phase of our algorithm consists of a route compression procedure which aims at reducing the total time duration of a route. In order to evaluate if a route is feasible, our heuristic relies on the computation of its schedule with respect to the time windows. All insertion and improvement procedures described so far aim at creating routes that leave the depot and arrive at each delivery location as soon as possible. It also has the advantage of giving a certain margin to the company dispatchers to compensate for unplanned execution problems. Since each delivery (and pause) has an associated time window, there is a risk of inducing unnecessary waiting times along the route that should be eliminated in order to minimize total route duration. For this reason, a route compression algorithm is included when scheduling the route.

Once the early schedule is established, the compression approach is applied when the total waiting time of the route is positive. The procedure is based on the forward time slack (FTS) concept, first described in Savelsbergh [26]. Without loss of generality, the early departure of a visit can be postponed by its FTS without causing any infeasibility in the route. Our compression procedure starts by computing F_0, the FTS of the depot, and sequentially moves the departure of each visit i until $F_i = 0$ or when all visits have been evaluated for postponement, whichever comes first.

In addition to the potential reduction in the total route duration, this compression procedure also has two positive side effects. First, the visit to the last customer is performed as soon as possible, which is usually appreciated by the customers and drivers. Second, the first visit is performed as late as possible, which besides being appreciated by the customers also helps drivers avoid the early morning traffic.

5 Computational experiments

In this section we describe the results of the computational experiments carried out to show the relevance of adequately planning drivers' pauses, and to evaluate the performance of the proposed mathematical formulation presented in Section 3 and of the heuristic presented in Section 4. First, we show that the VRPLB model cannot be solved to optimality even for very small instances. Second, we demonstrate that driver pauses should be considered in the optimization phase, and that neglecting this step, for example, by creating vehicle routes without pauses which are reinserted later, can lead to very poor solutions.

Our heuristic was implemented using VB.net. Numerical experiments were carried out on a desktop equipped with an Intel Core i7-3612QM running at 2.1 GHz and with 8 GB of RAM running MS Windows x64. The mixed integer linear programming formulation was implemented in C++ using CPLEX Concert Technology and the experiments were carried out on a desktop equipped with an Intel i7 running at 3.66 GHz and with 8 GB of RAM, under a Linux operating system.

We start by comparing the performance of the VRPLB formulation proposed in Section 3 to that of the heuristic presented in Section 4. We have generated 15 small instances based on a large real-life one obtained from our industrial partner. These instances contain from five to 20 delivery requests, and were solved by both methods. Table 1 presents average results, and shows that the branch-and-bound algorithm quickly becomes inefficient when the size of the instance increases (see column *Gap (%)*). In particular, we observe that after one hour of computing time, the branch-and-bound algorithm is not able to find optimal solutions, nor to yield reasonable gaps for instances containing 20 requests. Given that the real instances provided by our partner contain at least one hundred requests and can go up to 400 requests, it becomes clear that the proposed formulation is not suitable for real applications. Table 1 also shows that the computing time required by our heuristic remains extremely low when the size of the instances increases. Moreover, on these small instances for which the exact algorithm was able to obtain optimal solutions, our heuristic performed very well, being able to yield optimal solutions when these are known. This enables us to further evaluate the impact of the pauses in distribution problems.

Table 1: Comparison of the performance of the branch-and-bound algorithm and of the heuristic

# requests	B&B algorithm			Heuristic	
	Km	Gap (%)	Time (s)	Km	Time (s)
5	182.6	0.00	3.2	182.6	0.178
10	282.4	0.00	737.6	282.4	0.497
20	733.6	51.58	3600.0	709.8	1.365

We have shown that explicitly considering drivers pauses greatly increases the size of the model, so that only very small instances can be solved to optimality. The most intuitive workaround to adapt classical algorithms to obtain solutions for the VRPLB is to solve the problem in two phases as follows. First, one solves the problem without considering the pauses by using well known VRP algorithms. Then, one reintroduces the pause in the best possible position and adjust the delivery schedules accordingly. We now aim at showing how this two-phase approach for handling the pauses can lead to major solution infeasibilities. Moreover, we empirically demonstrate that the number of

infeasibilities rapidly increases when the percentage of deliveries with time windows increases, and also when time windows are tighter. A real instance with 209 deliveries provided by our industrial partner is used to support our experiments.

First, we removed all time windows from the instance. Then we randomly generated time windows for 10%, 20%, 30%, 40% and 50% of the deliveries. These time windows were randomly generated between 8:30 and 15:00. It is worth mentioning that since the drivers' pauses must be taken between 11:30 and 13:00, morning time windows are not impacted by the pause, but they still help shape the final solution. We have repeated this experiment twice, first considering 3-hours time windows, and then tighter 2-hours time windows. We solved these instances by the two-phase approach just described. To this end, we first set the pause time duration to zero and reduced the length of the day accordingly, i.e., drivers should return to the depot one hour earlier. The problem was then solved without pauses, using the heuristic described in Section 4. In the resulting solution, a pause with zero duration was included in each route. In the second phase, the pause length was set back to 60 minutes and the driver schedule after the pause was updated. We also tried to relocate the pause to all possible positions within its feasible time window to reduce infeasibilities as much as possible. If reinserting the pause leads to an infeasible solution, the position leading to the smallest number of violated time windows was selected. If many positions were feasible, the one which yielded the shortest trip duration was selected. A summary of these experiments is provided in Tables 2 and 3, where the column *% of TW* shows the percentages of deliveries having a time window, *Km* and *# of routes* shows the number of kilometers and the number of routes in the solution, respectively, the column *# of infeasible routes* provides the number of infeasible routes in the solution, and *# of infeasible customers* shows the total number of customers that are visited outside of their time windows. For our heuristic, we provide the length of the routes and number of vehicles used. Obviously, all routes yielded by our heuristic are feasible.

Table 2: Impact of neglecting drivers' pauses with three-hour time windows on an instance with 209 deliveries

% of TW	Two-phase approach				Our heuristic	
	Km	# of routes	# of infeasible routes	# of infeasible customers	Km	# of routes
0	2913	22	–	–	2913	22
10	3290	23	3	6	3433	23
20	3325	23	3	4	2891	23
30	3059	23	7	12	2871	23
40	3044	24	6	8	2957	24
50	3089	24	10	18	3353	25

Results reported in Table 2 clearly show the impact of neglecting driver pauses in the solution process when three-hour time windows are imposed. Even when only 30% of deliveries have time windows, seven routes out of 23 (30%) are infeasible, with 12 late deliveries (5.7%). When 50% of the requests contain time windows, 41% of the routes become infeasible and 18 deliveries are late (8.6%). If we consider that, on average, half of 209 deliveries are performed in the afternoon, this means in fact that 18 deliveries out of 104.5 are late, which increases the real percentage of late deliveries to 17.2%. The results of our heuristic, on the other hand, respect all time windows and typically present the same number of vehicles used. All the improvement procedures we have created are clearly capable of improving an initial solution, as can be observed by some cases in which our final solution is better than the (infeasible) solution without time windows. The performance of the heuristic will be evaluated later.

Table 3 presents results with tighter, two-hour, time windows. As expected, the results quickly deteriorate when time windows are not initially considered in the solution. For 30% and 50% of deliveries with time windows, the number of infeasible deliveries nearly doubled to 20 and 38 customers when comparing instances with 2- and 3-hour time windows. Our heuristic was still able to provide high quality solutions with the same number or few extra trucks. These results clearly demonstrate that driver pauses should be directly included in the resolution process and that not doing so yields a bad approximation which leads to numerous infeasibilities.

Table 3: Impact of drivers' pauses with two-hour time windows on an instance with 209 deliveries

% of TW	Two-phase approach				Our heuristic	
	Km	# of routes	# of infeasible routes	# of infeasible customers	Km	# of routes
0	2913	22	–	–	2913	22
10	2888	23	3	6	3180	23
20	3031	23	7	8	3154	23
30	2862	23	11	20	2992	23
40	3163	23	16	32	3609	24
50	3235	24	16	38	3490	27

Finally, in order to assess the performance of the heuristic on more challenging cases, we have solved two larger instances stemming from our industrial partner. These instances represent the real-life operations on two typical days and contain 290 and 382 deliveries. In the first one, the current software used by the company yields a solution with 3449 km while our heuristic returns 3232 km on a single run, and 2773 and 2725 km for 10 and 20 iterations, respectively, for a global improvement of 20% and a running time of only 20 minutes. For the second instance containing 382 deliveries, the

company solution requires 5138 km while the heuristic yields 4147, 3953 and 3856 km for its initial run, 10 and 20 iterations, for a global improvement 25%. The computing time was 46 minutes. For these two instances the ratios between the initial heuristic solution and the final one after 20 iterations are respectively 15% and 7%.

6 Conclusions

We have formally described a rich vehicle routing problem with pauses and time windows arising in the furniture delivery industry. We have proposed a mixed-integer linear programming formulation for this problem and showed that a branch-and-bound algorithm applied to it is impractical for real-life applications. We have also developed an efficient multi-start randomized local search heuristic which is capable of solving instances containing several hundred customers with time windows, several vehicles, and places the drivers pauses in the required time window. The results of our computational experiments confirm that one must solve the problem by taking into account the drivers' pauses explicitly in the optimization phase, or risk not being able to obtain feasible solutions for many situations. Our algorithm is now currently under evaluation by our industrial partner, already being integrated into their enterprise resource planning system.

Acknowledgments

This research was partially supported by Grants OPG 0293307, OPG 0172633 and 2014-05764 from the Canadian Natural Sciences and Engineering Research Council (NSERC) and by Engage Grant CGO 98069 from NSERC. This support is gratefully acknowledged. We also thank Louis Leclerc, Operations Manager, and Steve Thiboutot, Information Systems Manager, from Ameublement Tanguay, for their constant availability and comments, and for providing us with relevant data. We thank an associate editor and four anonymous referees for their valuables comments on a previous version of this paper.

References

[1] C. Archetti and M. Savelsbergh. The trip scheduling problem. *Transportation Science*, 43(4):417–431, 2009.

[2] C. Archetti, A. Hertz, and M. G. Speranza. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13(1):49–76, 2007.

[3] P. Avella, M. Boccia, and A. Sforza. Solving a fuel delivery problem by heuristic and exact approaches. *European Journal of Operational Research*, 152(1):170–179, 2004.

[4] A. Baltz, M. El Ouali, G. Jäger, V. Sauerland, and A. Srivastav. Exact and heuristic algorithms for the travelling salesman problem with multiple time windows and hotel selection. *Journal of the Operational Research Society*, 2014.

[5] A. M. Benjamin and J. E. Beasley. Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. *Computers & Operations Research*, 37(12):2270–2280, 2010.

[6] F. F. Boctor, J. Renaud, and F. Cornillier. Trip packing in petrol stations replenishment. *Omega*, 39(1):86–98, 2011.

[7] K. Buhrkal, A. Larsen, and S. Ropke. The waste collection vehicle routing problem with time windows in a city logistics context. *Procedia – Social and Behavioral Sciences*, 39:241–254, 2012. Seventh International Conference on City Logistics which was held on June 7- 9,2011, Mallorca, Spain.

[8] I.-M. Chao, B. L. Golden, and E. A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464–474, 1996.

[9] J. M. Day, P. D. Wright, T. Schoenherr, M. Venkataramanan, and K. Gaudette. Improving routing and scheduling decisions at a distributor of industrial gases. *Omega*, 37(1):227–237, 2009.

[10] K. C. Furman, J.-H. Song, G. R. Kocis, M. K. McDonald, and P. H. Warrick. Feedstock routing in the ExxonMobil downstream sector. *Interfaces*, 41(2): 149–163, 2011.

[11] A. Goel. Vehicle scheduling and routing with drivers' working hours. *Transportation Science*, 43(1):17–26, 2009.

[12] A. Goel. Truck driver scheduling in the european union. *Transportation Science*, 44(4):429–441, 2010.

[13] A. Goel and L. Kok. Truck driver scheduling in the United States. *Transportation Science*, 46(3):317–326, 2010.

[14] A. Goel and T. Vidal. Hours of service regulations in road freight transport: An optimization-based international assessment. *Transportation Science*, 48(3): 391–412, 2014.

[15] A. Goel, C. Archetti, and M. Savelsbergh. Truck driver scheduling in Australia. *Computers & Operations Research*, 39(5):1122–1132, 2012.

[16] B. L. Golden, A. A. Assad, and E. A. Wasil. Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, Monographs on Discrete Mathematics and Applications, pages 245–286. SIAM, Philadelphia, 2002.

[17] B.-I. Kim, S. Kim, and S. Sahoo. Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33(12):3624–3642, 2006.

[18] G. Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416, 2009.

[19] S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269, 1965.

[20] I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41(4):421–451, 1993.

[21] J. Privé, J. Renaud, F. F. Boctor, and G. Laporte. Solving a vehicle-routing problem arising in soft-drink distribution. *Journal of the Operational Research Society*, 57(9):1045–1052, 2006.

[22] J. Renaud, F. F. Boctor, and G. Laporte. An improved petal heuristic for the vehicle routeing problem. *Journal of the Operational Research Society*, 47(2): 329–336, 1996.

[23] P. P. Repoussis, D. C. Paraskevopoulos, G. Zobolas, C. D. Tarantilis, and G. Ioannou. A web-based decision support system for waste lube oils collection and recycling. *European Journal of Operational Research*, 195(3):676–700, 2009.

[24] Y. Rochat and F. Semet. A tabu search approach for delivering pet food and flour in Switzerland. *Journal of the Operational Research Society*, 45(11):1233–1246, 1994.

[25] S. Sahoo, S. Kim, B.-I. Kim, B. Kraas, and A. Popov Jr. Routing optimization for waste management. *Interfaces*, 35(1):24–36, 2005.

[26] M. W. P. Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, 4(2):146–154, 1992.

[27] C. D. Tarantilis and C. T. Kiranoudis. A meta-heuristic algorithm for the efficient distribution of perishable foods. *Journal of Food Engineering*, 50(1):1–9, 2001.

[28] C. D. Tarantilis and C. T. Kiranoudis. Distribution of fresh meat. *Journal of Food Engineering*, 51(1):85–91, 2002.

[29] M. F. Uzar and B. Çatay. Distribution planning of bulk lubricants at BP Turkey. *Omega*, 40(6):870–881, 2012.

[30] R.G van Anholt, L. C. Coelho, G. Laporte, and I. F. A. Vis. An inventory-routing problem with pickups and deliveries arising in the replenishment of automated teller machines. Technical Report CIRRELT-2013-71, Montreal, Canada, 2013.

[31] P. Vansteenwegen, W. Souffriau, and K. Sörensen. The travelling salesperson problem with hotel selection. *Journal of the Operational Research Society*, 63(2): 207–217, 2012.

[32] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3):658–673, 2014.