Les Cahiers du GERAD ISSN: 0711–2440

Compact routes for parcel delivery by postal services

A. Bretin, G. Desaulniers, L.-M. Rousseau

G–2021–04

February 2021

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2021
– Bibliothèque et Archives Canada, 2021

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2021
– Library and Archives Canada, 2021

Compact routes for parcel delivery by postal services

Alexis Bretin[a, b, c]

Guy Desaulniers[a, b]

Louis-Martin Rousseau[a, c]

[a] *Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montréal (Québec), Canada*

[b] *GERAD, Montréal (Québec), Canada*

[c] *CIRRELT, Montréal (Québec), Canada*

alexis.bretin@polymtl.ca
guy.desaulniers@gerad.ca
louis-martin.rousseau@cirrelt.net

February 2021
Les Cahiers du GERAD
G–2021–04

Abstract : In this paper, we introduce a new variant of the vehicle routing problem with time windows (VRPTW) that arises in parcel delivery by postal services. In addition to the standard VRPTW constraints and postal setting characteristics (low time window density and route duration constraints), the problem considered involves route compactness preferences and is called the VRPTW with route compactness (Comp-VRPTW). We measure the compactness of a route as the area of the convex hull of its visited customer locations and, to speed up its computation, we propose two approximate measures: the first frames the route in a rectangle, which can rotate; the second computes the area of a circle whose diameter is determined by the farthest pair of points in the route. This second approach also enables dealing with traveled time instead of geographical distance, which may be more relevant when the distances in the actual road network are far from being Euclidean (due to, e.g., highways, river crossings, or one-way streets). To solve the Comp-VRPTW, we adapt an existing branch-and-price-based large neighborhood search (LNS) heuristic initially designed for the VRPTW. To evaluate the tradeoff between route compactness and the total travel cost, we report computational results obtained on academic Comp-VRPTW instances and on instances derived from a real one provided by our industrial partner. These results show that the proposed measures can greatly improve route compactness at the expense of an increase of the LNS heuristic computational times and of the travel costs for the academic instances that are highly time-constrained.

Keywords: Vehicle routing problem with time windows, parcel delivery, postal services, route compactness, approximate measures

Acknowledgements: We would like to thank Charles Fleurent and his team at GIRO Inc. for suggesting this problem to us and providing a real-life dataset. We gratefully acknowledge the financial support of GIRO Inc. and the Natural Sciences and Engineering Research Council of Canada under the grant RDCPJ 520349-17.

1 Introduction

This paper deals with parcel delivery performed by postal services, where parcels are normally handled separately from mail and delivered by trucks. Even if parcel delivery by drones is getting increased attention in the literature (see, e.g., Agatz et al., 2018; Murray and Raj, 2020), we focus on traditional truck deliveries because drones are not yet broadly deployed for this task. The problem that we consider belongs to the class of the vehicle routing problem (VRP). Introduced by Dantzig and Ramser (1959), the VRP is defined as computing the routes of a fleet of identical capacitated vehicles, originally located at a central depot, such that each customer of a given set is visited exactly once by a vehicle to satisfy its demand, the total load delivered along a route does not exceed vehicle capacity, and a cost function often related to the total distance traveled is minimized. One of its most common variant is the VRP with time windows (VRPTW) which restricts the service at a customer to start within a predefined time interval. For parcel delivery, not all deliveries are subject to time window restrictions because they are present mainly for commercial customers or to respect fast-delivery contracts (*e.g.*, within 24 or 48 hours).

In the scientific literature, many solution algorithms have been proposed to solve the VRP and its variants. For many of them, the leading exact algorithms are branch-price-and-cut algorithms as surveyed by Costa et al. (2019). Heuristics are, however, unavoidable to solve large instances. Some of the most performing ones are the hybrid genetic algorithm of Vidal et al. (2013) and the memetic algorithm of Nagata et al. (2010). For the VRPTW, the branch-and-price-based large neighborhood search (LNS) algorithm of Prescott-Gagnon et al. (2009) also produced satisfying computational results (see Desaulniers et al., 2014). Our work is based on this matheuristic.

In most parcel delivery companies, especially postal agencies, the objective of the parcel delivery problem considered is to find a solution that offers a good compromise between the usual routing cost, which is proportional to the distance traveled (or, sometimes, the time spent traveling), and the compactness of the routes which can be measured as discussed below. The compactness criterion is motivated by two main factors: territory knowledge and route acceptability. Indeed, when routes are compact, it is easier to assign to each driver a route that is restricted to a territory (neighborhood) that is well known by the driver, thus, increasing his/her effectiveness and reducing the driver's stress. Also, as indicated to us by our industrial partner GIRO (which develops route optimization software for postal services), compact routes are better accepted by the route planners and the drivers for the following reasons. On the one hand, compact routes remain more or less in the same neighborhood and can, thus, be assigned more easily by the planners to the drivers according to the drivers' knowledge of specific neighborhoods. On the other hand, compact solutions tend to reduce route overlapping, lowering the chances that several drivers are in the same area at the same time, which has been reported as frustrating for the drivers who believe that intersecting routes are inefficient. Finally, compact routes are also appreciated by the drivers of some companies that reward them for each parcel delivered in hand because compactness favors short returns to absent customers.

This paper focuses on the concept of route compactness which is not well referenced in the literature (see Section 2.1). We propose to measure the compactness of a route as the area of the convex hull of the locations of its visited customers (thus, excluding the depot). For the sake of conciseness, this convex hull is called the convex hull of the route. Because computing the area of a polygonal convex hull may be too time-consuming to be performed within an optimization algorithm, we introduce two approaches to approximate the convex hull of a route. The first one, purely geographical, is a rectangular approximation. The second one, which is more easily adaptable to handle road network specificities, is a disk approximation where the disk diameter may be defined not only as the Euclidean distance between two customer locations but also, for example, as the travel time between two locations.

To assess the tradeoff between routing costs and route compactness, we adapt the branch-and-price-based LNS algorithm of Prescott-Gagnon et al. (2009) such that it can handle route compactness penalties as well as other features related to route duration as discussed below. Note that our intent is not to devise the most efficient algorithm for solving the problem at hand, but rather to have an algorithm that is applicable to compare the proposed approximate compactness measures. Our computational experiments, performed on academic and on instances derived from a real-world dataset, allow such a comparison and to evaluate the additional computational burden for dealing with route compactness in this LNS heuristic.

Our contribution is thus threefold. First, we introduce a new variant of the VRPTW that we call the VRPTW with route compactness and denote Comp-VRPTW. Second, we propose and study different route compactness measures. Third, we modify an existing heuristic so that it can handle these route compactness measures.

The rest of this paper is structured as follows. In Section 2, we define the parcel delivery problem considered *i.e.*, the Comp-VRPTW, introduce the proposed route compactness measures, and provide a mathematical formulation for the Comp-VRPTW. In Section 3, we describe the enhanced branch-and-price-based LNS algorithm. Computational results on academic and industry-derived instances are reported in Sections 4 and 5, respectively. Finally, a short conclusion is drawn in Section 6.

2 Problem definition and mathematical formulation

In postal services, the amount of customers to visit in a route is usually quite large (between 50 and 100), whereas the weight or volume of every single delivery is rather limited. For this reason, the routes are typically time-constrained but not necessarily capacity-constrained. This time constraint is referred to as the route duration (RD) constraint. Note that route duration is closely related to the distance traveled, but also to the set of customers visited, because the service time may vary from one customer to another. For a given route, it is usually computed as the sum of the total travel time, service time, and waiting time if any.

The Comp-VRPTW can be stated as follows. Given a set of customers V^C to service exactly once and a homogeneous fleet of capacitated vehicles L, the problem consists in designing compact feasible routes such that a weighted sum of the routing costs and the compactness costs is minimized. In the following, we detail route feasibility and the cost structure.

Each customer $i \in V^C$ is located at coordinates (X_i, Y_i). The service at some customers $i \in V^C$ must start within a time window $[\underline{w}_i, \bar{w}_i]$. For the other customers, we assume that they are associated with an unconstraining time window $[\underline{w}_i, \bar{w}_i]$, where $\underline{w}_i = 0$ and $\bar{w}_i$ is equal to a large value W. Note that a vehicle can arrive earlier than $\underline{w}_i$ at customer i but must wait until $\underline{w}_i$ before starting service. A service time s_i and a demand q_i are associated with each customer $i \in V^C$. All routes start and end at the same depot which is denoted d. The departure time of a route is not fixed but must be greater than or equal to 0. Let $V = V^C \cup \{d\}$ be the set of all customer and depot locations. With each pair of distinct locations $(i,j) \in V^2$, $i \neq j$, are associated a travel time t_{ij} as well as a travel cost c_{ij} which is generally proportional to t_{ij}.

A feasible route r is given by a sequence of locations $(v_0, v_1, \ldots, v_p)$ with $p \geq 2$ such that v_0 and v_p represent the depot at the beginning and the end of the route, respectively, $v_i \in V^C$, $i \in \{1, \ldots, p-1\}$, and $v_i \neq v_j$ for all $i, j \in \{1, \ldots, p-1\}$, $i \neq j$. It is associated with a schedule $(T_{v_0}, T_{v_1}, \ldots, T_{v_p})$, where T_{v_0} and T_{v_p} denote the route starting and ending times, respectively, and T_{v_i} the start of service time at customer v_i, $i \in \{1, \ldots, p-1\}$. This schedule must respect the customer time windows, *i.e.,* it must satisfy the following conditions:

$$T_{v_0} \geq 0 \tag{1}$$

$$T_{v_{i-1}} + s_{v_i} + t_{v_{i-1},v_i} \leq T_{v_i}, \qquad \forall i \in \{1, \ldots, p-1\} \tag{2}$$

$$T_{v_i} \in [\underline{w}_i, \bar{w}_i], \qquad \forall i \in \{1, \ldots, p-1\}, \tag{3}$$

with $s_{v_0} = 0$. When the times $(T_{v_0}, T_{v_1}, \ldots, T_{v_p})$ are seen as variables, the (minimal) duration U_r of route r is equal to the optimal value of the linear program:

$$U_r = \min \quad T_{v_p} - T_{v_0} \tag{4}$$

$$\text{subject to:} \quad (1)\text{–}(3).$$

To avoid excessive driver fatigue, this duration must not exceed U^{max}, a user-defined parameter which generally depends on the country's labor legislation and the drivers' union rules. Finally, when the capacity of a vehicle Q^{max} must be taken into account, the total demand of the visited customers along route r must not exceed this capacity, *i.e.*, $\sum_{i=1}^{p-1} q_i \leq Q^{max}$.

The cost c_r of route r is given by

$$c_r = \tau_r + \lambda\kappa_r, \tag{5}$$

where τ_r and κ_r are the routing cost and the compactness measure of route r, respectively, and $\lambda \geq 0$ is a weight that can be adjusted to put more or less emphasis on the compactness criterion. The routing cost τ_r is given by $\tau_r = F + \sum_{i=0}^{p-1} c_{v_i,v_{i+1}}$, where F is a fixed cost sufficiently large to ensure that the number of vehicles used is minimized.

In the following, we discuss different compactness measures that can be used for κ_r (Section 2.1) and provide a mathematical formulation for the Comp-VRPTW (Section 2.2).

2.1 Compactness measures

The scientific literature on vehicle route compactness is still scarce. For waste collection, Kim et al. (2006) introduced a shape metric to measure the compactness of a route. This measure is computed, in practice, as the sum of the Euclidian distances between the locations visited in the route and its gravity center. It has the disadvantage of considering individually each visited location which may result in an overestimated measure when several locations are close together but relatively far from the gravity center. In addition, to design better solutions, these authors consider how the route convex hulls overlap in the heuristic they propose. Trying to identify which characteristics favor good VRP solutions, Arnold and Sörensen (2019) defined two new measures of compactness. The first sums over all visited customers in a route the Euclidian distances between a customer and the straight line passing through the depot and the gravity center of the route. The second rather sums the angles between this line and the line linking the depot and a customer. These measures tend to generate petal-shaped routes around the depot, which is not necessarily the most desired pattern in postal services because the neighborhoods nearby the depot are covered by too many drivers. Another definition of compactness was proposed by Ríos-Mercado and Fernández (2009) for a sales territory design problem which is defined on a graph. The compactness of a territory composed of a subset of nodes in this graph is computed as the maximum distance between each node in this subset and a center that corresponds to one of the subset nodes. As in Kim et al. (2006), the choice of the center is a decision variable but it is limited to a discrete set of possibilities.

To take into account a route compactness measure while solving the Comp-VRPTW, this measure must meet the following two criteria:

1. It should be based on the customer set only and not be sequence-dependent;
2. It should be relatively fast to compute or at least to update.

Criterion 1 captures the idea of minimizing some geometric area in which the driver has to work, without penalizing any detour inside this area that may be caused, for instance, by a customer with a tight time window. In other words, a customer located inside the convex hull of a route should belong to this route. Furthermore, according to our industrial partner, the depot location should not be taken into consideration when measuring the compactness of a route. Criterion 2 stems from the application context. Parcel delivery in postal services is very dynamic, and routes from one day to another may be very different and should be recomputed every night in a relatively narrow time slot. For this reason, the computational impact of considering compactness should be limited.

Measuring the compactness of a route as the area of the convex hull of its visited customers satisfies the first criterion, but not the second. Therefore, after presenting this time-consuming measure first, we introduce three approximate measures (the last one combining the other two) which are less time-consuming.

2.1.1 Exact computation

In this section, we briefly explain how the area of the convex hull of a route can be computed exactly. For our computational experiments, we performed this computation *a posteriori* on the final solution to serve as a comparison for evaluating the proposed approximate measures.

To compute this area, the convex hull has to be identified first. The most efficient ways to proceed are with the algorithm of Graham (1972) in $\mathcal{O}(n \log n)$ or that of Chan Chan (1996) in $\mathcal{O}(n \log h)$, where n is the number of customers in the route and h the number of these customers defining the convex hull. Because h is unknown in practice, Chan's algorithm starts by choosing a parameter m and is successful if and only if $m \geq h$ (the complexity is then $\mathcal{O}(n \log m)$). In the worst case, $m = n$ and we assume in the following that the worst-case complexity of identifying the convex hull of a route is given by $\mathcal{O}(n \log n)$.

Once the convex hull is identified, the computation of its area A is quite straightforward. Let N be the number of vertices of the corresponding polygon and assume that these vertices (X_1, Y_1) to (X_N, Y_N) are ordered, following the boundary of the polygon, either clockwise or counterclockwise. The area is then given by:

$$A = \frac{1}{2} \left| \sum_{i=1}^{N} X_i\, Y_{i+1} - X_{i+1}\, Y_i \right|,$$

where $(X_{N+1}, Y_{N+1}) = (X_1, Y_1)$. Thus, this computation requires $\mathcal{O}(n)$ operations.

In the following, we denote by κ_r^E the area A computed for a route r and call it the exact compactness measure of route r.

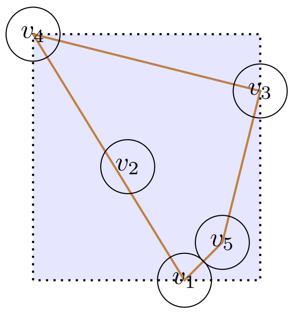
2.1.2 Rectangle approximation

Our first suggestion is to approximate the convex hull of a route $r = (v_0, v_1, \ldots, v_p)$ with a rectangle that contains all the visited customers in r and whose sides are parallel to the axes of the geographical coordinates. Let

$$x_r^{Min} = \min_{i \in \{1,\ldots,p-1\}} X_{v_i}, \qquad x_r^{Max} = \max_{i \in \{1,\ldots,p-1\}} X_{v_i},$$
$$y_r^{Min} = \min_{i \in \{1,\ldots,p-1\}} Y_{v_i}, \qquad y_r^{Max} = \max_{i \in \{1,\ldots,p-1\}} Y_{v_i},$$

be the minimum and maximum values of the X- and Y-coordinates of the visited customers. The area A of the rectangles defined by the points (x_r^{Min}, y_r^{Min}), (x_r^{Min}, y_r^{Max}), (x_r^{Max}, y_r^{Min}), and (x_r^{Max}, y_r^{Max}) is given by $A = (x_r^{Max} - x_r^{Min})(y_r^{Max} - y_r^{Min})$. However, to avoid the critical case where all customers are colinear on a line parallel to an axis, we introduce a minimum width and length μ (*e.g.*, 1% of the largest dimension) and propose the following rectangle compactness measure, denoted κ_r^R:

$$\kappa_r^R = \min\{\mu, x_r^{Max} - x_r^{Min}\} \times \min\{\mu, y_r^{Max} - y_r^{Min}\}.$$



Points	(X_i, Y_i)
v_1	(0,0)
v_2	(-3,6)
v_3	(4,10)
v_4	(-8,13)
v_5	(2,2)

Figure 1: A rectangle approximating the convex hull of a route

In Figure 1, we present an example with five customers for which $x_r^{Min} = -8$, $x_r^{Max} = 4$, $y_r^{Min} = 0$, and $y_r^{Max} = 13$, yielding $\kappa_r^R = 12 \times 13 = 156$. Graphically, one can see that the rectangle is a relatively poor approximation of the route convex hull (in brown) because there is a significantly large empty space in the lower-left corner of the rectangle. Allowing rotated rectangles could potentially improve the approximation. However, for computational efficiency, we consider only a subset of the possible rotation angles. Furthermore, instead of rotating the rectangles throughout the execution of the algorithm, we precompute the rotated coordinates of each customer once at the beginning of the algorithm for each considered angle, allowing a fast computation of the area of the rectangles using only the minimum and maximum functions as described above. Note that rotating all the points does not change their relative location one to another, so the area remains consistent. For a rotation angle θ, the computation of the new coordinates $(X_{v_i}^\theta, Y_{v_i}^\theta)$ for a customer v_i is performed using the following formula:

$$\begin{pmatrix} X_{v_i}^\theta \\ Y_{v_i}^\theta \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} X_{v_i} \\ Y_{v_i} \end{pmatrix}$$

where $(X_{v_i}, Y_{v_i}) = (X_{v_i}^0, Y_{v_i}^0)$ are the original coordinates of the point.

Let Θ be the subset of rotation angles considered, *e.g.*, $\Theta = \{0, 15, 30, 45, 60, 75\}$ (in degrees). Because of the symmetry between the rectangles, one can easily observe that it is sufficient to select only angles in the semi-open interval $[0, 90[$. Then, for each angle $\theta \in \Theta$ and corresponding dataset of coordinates $(X_{v_i}^\theta, Y_{v_i}^\theta)$, $i \in \{1, \dots, p-1\}$, we compute as above the minimum and maximum values of the X^θ- and Y^θ-coordinates:

$$x_r^{Min}(\theta) = \min_{i \in \{1,\dots,p-1\}} X_{v_i}^\theta, \qquad x_r^{Max}(\theta) = \max_{i \in \{1,\dots,p-1\}} X_{v_i}^\theta,$$
$$y_r^{Min}(\theta) = \min_{i \in \{1,\dots,p-1\}} Y_{v_i}^\theta, \qquad y_r^{Max}(\theta) = \max_{i \in \{1,\dots,p-1\}} Y_{v_i}^\theta,$$

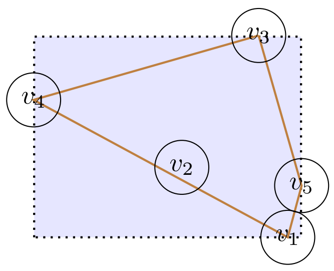
which are used to compute a rectangle compactness measure associated with θ:

$$\kappa_r^R(\theta) = \min\{\mu, x_r^{Max}(\theta) - x_r^{Min}(\theta)\} \times \min\{\mu, y_r^{Max}(\theta) - y_r^{Min}(\theta)\}.$$

Finally, to determine the final rectangle compactness measure of route r. we take the minimum of the measures computed over all considered angles, *i.e.*,

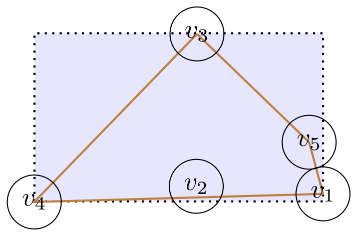
$$\kappa_r^R = \min_{\theta \in \Theta} \kappa_r^R(\theta).$$

Let us return to the example in Figure 1 which provided $\kappa_r^R(0) = 156$. If we choose $\Theta = \{0, 30, 60\}$, we also have to consider the rectangles drawn in Figures 2 and 3 which yield $\kappa_r^R(30) = 14.16 \times 10.66 = 150.95$ and $\kappa_r^R(60) = 15.26 \times 8.89 = 135.66$. Finally, we obtain $\kappa_r^R = \min\{156, 150.95, 135.66\} = 135.66$, a significant improvement over the no-rotation value $\kappa_r^R(0) = 156$. As it will be discussed in Section 4.2, the more angles in set Θ, the better the approximation. However, considering more angles increases computational times and, therefore, a compromise between measure precision and computational time must be achieved.



Points	(X_i^{30}, Y_i^{30})
v_1	(0,0)
v_2	(-5.60,3.70)
v_3	(-1.54,10.66)
v_4	(-13.43,7.26)
v_5	(0.73,2.73)

Figure 2: Rotated rectangle approximation ($\theta = 30°$)



Points	(X_i^{60}, Y_i^{60})
v_1	(0,0)
v_2	(-6.70,0.40)
v_3	(-6.66,8.46)
v_4	(-15.26,-0.43)
v_5	(-0.73,2.73)

Figure 3: Rotated rectangle approximation ($\theta = 60°$)

2.1.3 Disk approximation

As a second approximation of the convex hull of a route $r = (v_0, v_1, \dots, v_p)$, we propose to use a disk whose diameter D_r is given by the maximum Euclidian distance between two customers visited in r and whose boundary passes by two customers yielding this maximum distance. More precisely, let $V_r = \{v_1, \dots, v_{p-1}\}$ be the set of visited customers in r and e_{v_i,v_j} the Euclidian distance between customers v_i and v_j in V_r. Then, $D_r = \max_{(v_i,v_j)\in V_r^2} e_{v_i,v_j}$. The corresponding disk compactness measure, denoted κ_r^D, is then given by the disk area:

$$\kappa_r^D = \left(\frac{D_r}{2}\right)^2 \pi.$$

For the example in Figure 1, the most distant customers are v_1 and v_4, yielding a diameter of $\sqrt{233}$ and a disk compactness measure of $\kappa_r^D = (\sqrt{233}/2)^2\,\pi = 183.0$. A graphical representation of the corresponding disk is given in Figure 4, which shows that the disk does not necessarily contain all visited customers and, therefore, the whole route convex hull. Nevertheless, it is easy to see that, when visiting the same number of customers, routes with smaller disk areas are, in general, more compact than those with larger areas.

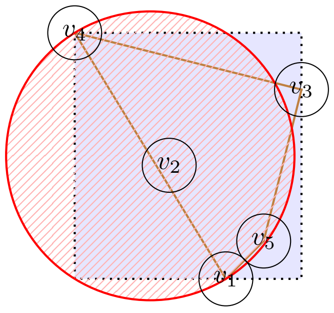


Figure 4: Disk and rectangle approximations

One can notice that minimizing the area of a single disk is equivalent to minimizing the diameter of this disk. However, this equivalence does not hold when seeking to minimize the sum of several disk areas (induced by multiple routes like in the Comp-VRPTW). Indeed, consider an example with two disks and two distinct solutions with an equal sum of the diameters: diameters 2 and 8 for the first solution and diameters 4 and 6 for the second. Computing the sums of the corresponding disk areas yields unequal values, namely, $\pi + 16\pi = 17\pi$ and $4\pi + 9\pi = 13\pi$. Nevertheless, using the diameter directly as a compactness measure, *i.e.*, setting $\kappa_r = D_r$, may be suitable when interesting alternatives to the Euclidean distance make sense. In fact, in practice, when travel times are calculated according to a certain road network, including some traffic regulations (one-way streets, prohibited left turns, ...) and geographical constraints (rivers with few bridges, mountains with or without tunnels, ...), computing the route compactness with real network distances (shortest path lengths) or travel times may be more appropriate than with distances as crow flies. However, when doing so, any geographical area comparison becomes meaningless.

2.1.4 Hybrid route compactness measure

The quality of the two approximations introduced above depends on the shape of the route. As the shape of the routes considered in the proposed algorithm may vary significantly, mixing both rectangle disk compactness measures should lessen and rectify low-quality fits. Therefore, we define κ_r^H, the hybrid compactness measure of a route r as a convex combination of κ_r^R and κ_r^D, *i.e.*,:

$$\kappa_r^H = \omega^R \kappa_r^R + \omega^D \kappa_r^D,$$

where $\omega^R, \omega^D \geq 0$ and $\omega^R + \omega^D = 1$. These parameters are user-defined weights that can be adjusted to favor one measure over the other. Note that this hybrid measure encompasses the two previous measures. Indeed, setting $\omega^R = 1$ and $\omega^D = 0$ yields $\kappa_r^H = \kappa_r^R$, whereas setting $\omega^R = 0$ and $\omega^D = 1$ yields $\kappa_r^H = \kappa_r^D$.

2.2 Mathematical formulation of the Comp-VRPTW

Let Ω be the set of all feasible routes. According to (5), the cost of a route $r \in \Omega$ is expressed as $c_r = \tau_r + \lambda \kappa_r$. In theory, we would like to set $\kappa_r = \kappa_r^E$. However, given the computational complexity of considering κ_r^E during the solution process, we rather use $\kappa_r = \kappa_r^H$ to yield $c_r = \tau_r + \lambda^R \kappa_r^R + \lambda^D \kappa_r^D$, where $\lambda^R = \lambda \omega^R$ and $\lambda^D = \lambda \omega^D$. Let a_{ir}, $i \in V^C$, $r \in \Omega$, be a binary parameter equal to 1 if route r covers customer i and to 0 otherwise. Finally, with each route $r \in \Omega$, we associate a binary variable α_r that takes value 1 if route r belongs to the solution and to 0 otherwise.

With this notation, the Comp-VRPTW that we consider can be formulated as follows:

$$\min \quad \sum_{r \in \Omega} c_r \alpha_r \tag{6}$$

$$\text{subject to:} \quad \sum_{r \in \Omega} a_{ir} \alpha_r = 1, \qquad \forall i \in V^C \tag{7}$$

$$\sum_{r \in \Omega} \alpha_r \leq m_{ub}, \tag{8}$$

$$\alpha_r \in \{0, 1\}, \qquad \forall r \in \Omega. \tag{9}$$

Objective function (6) aims at minimizing a weighted sum of the total routing and compactness costs. Constraints (7) ensure customer coverage, while constraint (8) restricts the number of routes used in the solution to $m_{ub} = |L|$, the maximum number of vehicles available in fleet L. Binary requirements on the decision variables are imposed through constraints (9).

3 Solution algorithm

To solve the Comp-VRPTW, we propose to adapt the branch-and-price-based LNS heuristic developed by Prescott-Gagnon et al. (2009) for tackling the VRPTW. Starting from an initial solution, this algorithm proceeds in two phases: the first one aims for a vehicle number reduction (VNR) which is aligned with the primary objective of the Comp-VRPTW (minimizing the number of vehicles assuming that the vehicle fixed cost F is sufficiently large), whereas the second seeks to achieve a traveled distance reduction (TDR). In the VNR phase, the upper bound m_{ub} defining the right-hand side of constraint (8) is first set to the number of vehicles used in the initial solution minus one. If a feasible solution can be found within a certain number of LNS iterations I_{max}^{VNR}, m_{ub} is decremented again by one and new LNS iterations are performed. This process continues until no feasible solution can be found for a given bound or m_{ub} reaches a pre-computed lower bound m_{lb}. In the former case, m_{ub} is increased by one before moving on to the TDR phase. Note that with this strategy, there is no need to consider the fixed cost F in both phases. Furthermore, given the additional computational burden of handling the compactness costs, they are not taken into account in the VNR phase.

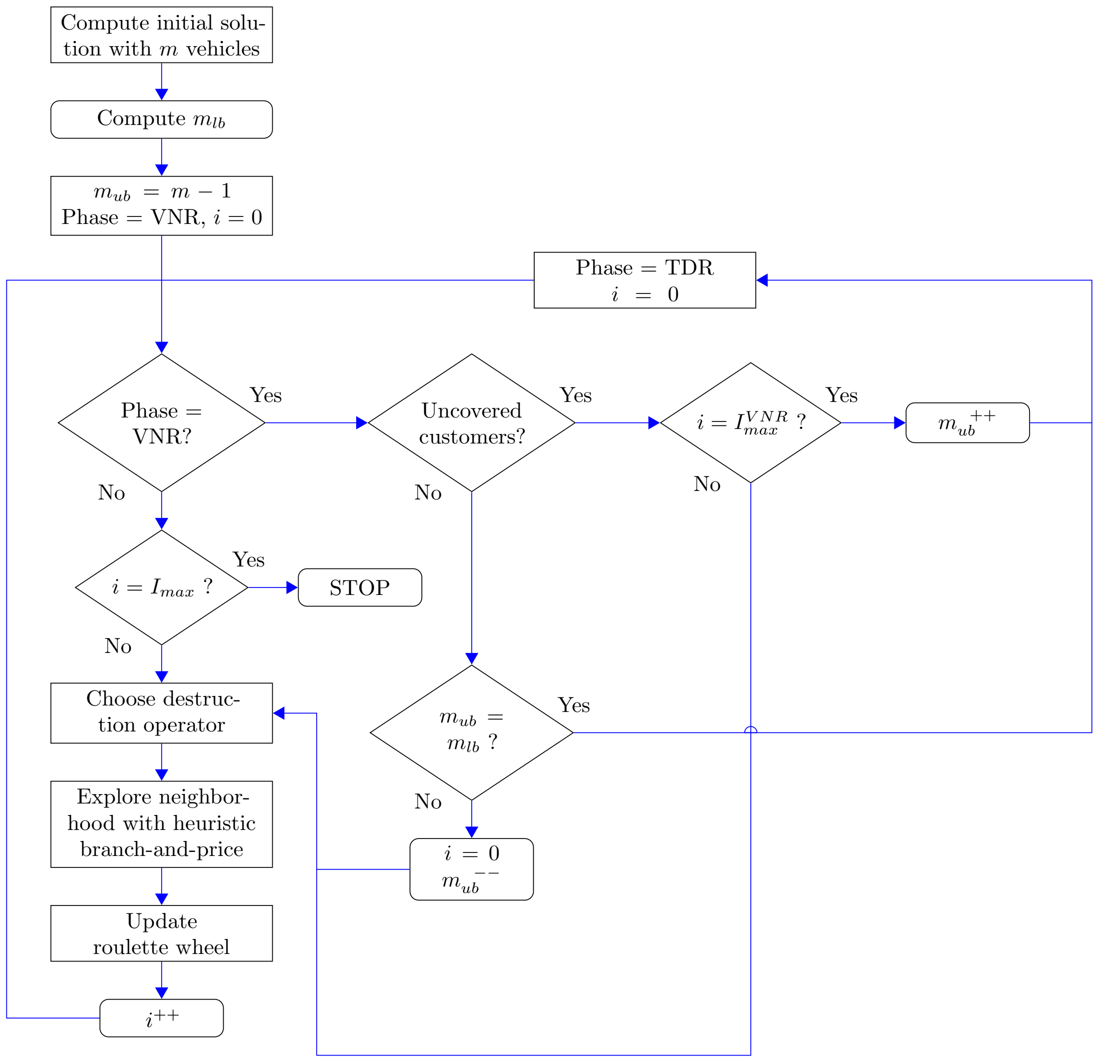


Figure 5: Algorithm flowchart

The solution process applied in both phases is roughly the same, but differs by the parameter setting. The general behavior of the LNS algorithm is presented in Figure 5. The main loop of the algorithm consists of the following sequence of steps. Starting from the current solution, we partially destroy it using a destruction operator, which is chosen randomly from a predefined set of four operators using a roulette-wheel procedure similar to that of Pisinger and Ropke (2007). This procedure favors operators that have been successful in the recent iterations, *i.e.*, their application yielded an improved solution at the end of the LNS iteration. The chosen destruction operator removes customers from the routes in the current solution, leaving fixed partial routes.

After destruction, a branch-and-price heuristic applied to model (6)–(9) is called to rebuild a solution taking into account the fixed partial routes. The branch-and-price heuristic applies heuristic column generation to solve linear relaxations and a diving branching strategy to derive an integer solution. Column generation is an iterative algorithm that solves at each iteration a restricted master problem, i.e., the linear relaxation of (6)–(9) restricted to a subset of its variables, and a subproblem which finds negative reduced cost routes to add to the restricted master problem. Usually, column generation stops when there are no more negative reduced cost routes. In our case and as proposed by Prescott-Gagnon et al. (2009), we solve the subproblem heuristically using the tabu search heuristic of Desaulniers et al. (2008), possibly failing to find a negative reduced cost route when some exist. This heuristic offers a lot of flexibility (in particular to handle the compactness costs) and, typically, yields shorter computational times. In fact, at each column generation iteration, it is applied in a multi-start fashion, starting each time from a column in the current basis of the restricted master problem, for a very limited number of tabu search iterations. Only two types of moves are considered, namely, customer insertion and customer removal. During the TDR phase where the compactness costs are taken into account, the evaluation of the possible insertion and removal moves must consider their impact on the compactness measure of the routes.

The branch-and-price heuristic generally yields a new solution that becomes the current solution even if it does not improve over the previous one. It may also happen that this heuristic fails to find any solution, in which the previous solution is kept as the current one. The algorithm moves on to update the roulette-wheel statistics before starting a new iteration. The other steps in Figure 5 are invoked to initialize the LNS heuristic and to determine when to lower the m_{ub} bound, when to switch from the VNR to the TDR phase, and when to stop the LNS heuristic. Further details about this branch-and-price-based LNS heuristic can be found in Prescott-Gagnon et al. (2009).

Below, we focus on the description of the algorithm adaptations that were necessary to tackle the Comp-VRPTW. More precisely, we discuss how we compute the initial solution and how the tabu search column generator handles the route duration constraint and the compactness costs.

3.1 Initial solution

Finding a good initial solution helps to reduce the computational time of the VNR phase, but it should not be too much time consuming. To favor compactness minimization in the TDR phase, we generate this solution by first splitting the set of customers into disjoint subsets according to geographical boundaries such that each subset contains approximately the same number of customers. The number of subsets depends on the instance size. A graphical example is presented in Figure 6 for an instance with N customers that have to be partitioned into four subsets (if $N \bmod 4 \neq 0$ the number of customers in a subset is equal to $\lfloor N/4 \rfloor$ or $\lceil N/4 \rceil$). Each customer subset yields a sub-instance which is solved by a simple VRPTW constraint programming algorithm, restricted to a tight time limit (*e.g.*, 10 seconds). The constraint programming model used is described in Appendix A. As the sub-instances are completely independent, the computation of the initial solution can be performed in parallel (this is not the case for our tests). The impact of the quality of the initial solution is discussed in Section 4.4.

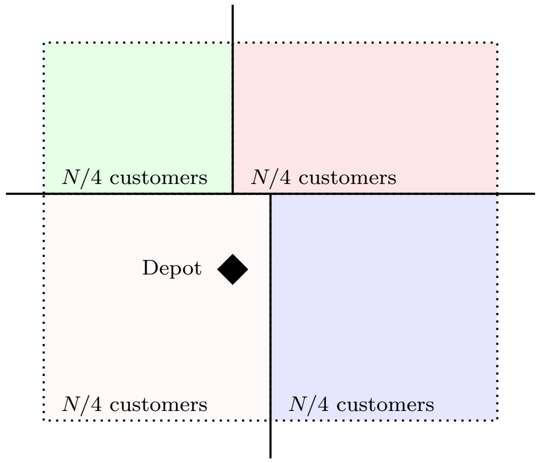


Figure 6: Example of an instance splitting for finding an initial solution

3.2 Adapted tabu search route generator

As mentioned above, some adaptations to the tabu search column (route) generator are required to be able to solve the Comp-VRPTW with the branch-and-price-based LNS framework of Prescott-Gagnon et al. (2009). They aim at computing the duration of a route and measuring its compactness. We recall that only two move types are applied in this tabu search heuristic, namely, customer insertion and customer removal.

3.2.1 Route duration computation

In the Comp-VRPTW, the routes are subject to a maximum duration constraint, *i.e.*, the duration U_r of each route $r \in \Omega$ must be such that $U_r \leq U^{max}$. In the tabu search algorithm, there is no need to verify this constraint when removing a customer from a feasible route r. On the other hand, a feasibility check must be performed when inserting a customer in r to yield a new route r' because $U_{r'} \geq U_r$. Recall that the duration U_r of a route r can be computed as the optimal value of the linear program (1)–(4). However, solving a linear program might be prohibitively time-consuming in a tabu search algorithm. We rather use a simple procedure with complexity $\mathcal{O}(|V_r|)$, where V_r is the subset of customers visited in route r. To understand the idea behind this procedure, we first discuss the example of a route $r = (v_0, v_1, \ldots, v_4, v_5)$ with four customers in Table 1. We assume that $s_{v_i} = 2$ for all $i = 1, \ldots, 4$, and $t_{v_{i-1},v_i} = 3$ for all $i = 1, \ldots, 5$. The first row in this table displays the time window associated with each location where $W = 50$. The second row presents a feasible schedule $(T_{v_0}, T_{v_1}, \ldots, T_{v_5})$ for route r where the service at each customer starts as early as possible and the departure and arrival times at the depot are also as early as possible. With this schedule, the duration of route r is equal to $T_{v_5} - T_{v_0} = 36$. Finally, the last row provides a schedule $(T^*_{v_0}, T^*_{v_1}, \ldots, T^*_{v_5})$ that minimizes route duration, yielding $U_r = T^*_{v_5} - T^*_{v_0} = 29$. To obtain this schedule, the departure time from the depot was shifted forward by 7 compared to the first schedule, where 7 is equal to the slack between $\bar{w}_{v_1}$ and T_{v_1}. This slack was compressed to save waiting time before servicing customers v_2 and v_4 and to reduce the duration of the route. Note that the times remain the same for the last two locations v_4 and v_5 because it is not possible to further move ahead the departure time from the depot without violating the time window of a customer before v_4.

Table 1: Two schedules for the same route with four customers

	v_0	v_1	v_2	v_3	v_4	v_5
$[\underline{w}_{v_i}, \bar{w}_{v_i}]$	$[0, 50]$	$[3, 10]$	$[12, 25]$	$[0, 50]$	$[31, 35]$	$[0, 50]$
T_{v_i}	0	3	12	17	31	36
$T^*_{v_i}$	7	10	15	20	31	36

This example illustrates for a simple case that the minimal duration of a route can be computed by assigning first the earliest service start times to the customers, while keeping track of the slack that can be used to shift forward the time of departure from the depot to avoid waiting at a customer. This is the rationale behind the procedure that we use to compute the duration of a route. The pseudo-code of this procedure is given in Algorithm 1, where S stores the current slack time available. For each customer v_i, it first computes its earliest arrival time in Step 5. If this arrival occurs before the opening of the customer time window (Step 6), we try to shift forward the departure time from the depot (Step 7) to avoid all waiting time if possible or by the amount of slack time available otherwise. Note that the times associated with the customers $v_1, v_2, \ldots, v_{i-1}$ are also shifted but there is no need to make those computations if we only search for the route duration. Next, the slack time is reduced by the amount consumed (Step 8) and the service start time at v_i is set to $\underline{w}_{v_i}$ (Step 9). Finally, the slack time is updated with respect to the time window upper bound $\bar{w}_{v_i}$ in Step 10 before moving on to the next visited location in route r.

Algorithm 1 Route duration computation

```
1:  Input: Route $r = (v_0, v_1, \ldots, v_p)$
2:  $T_{v_0} \leftarrow 0$
3:  $S \leftarrow W$
4:  for $i = 1, \ldots, p$ do
5:      $T_{v_i} \leftarrow T_{v_{i-1}} + s_{v_{i-1}} + t_{v_{i-1},v_i}$
6:      if $T_{v_i} < \underline{w}_{v_i}$ then
7:          $T_{v_0} \leftarrow T_{v_i} + \min\{S, \underline{w}_{v_i} - T_{v_i}\}$
8:          $S \leftarrow S - \min\{S, \underline{w}_{v_i} - T_{v_i}\}$
9:          $T_{v_i} \leftarrow \underline{w}_{v_i}$
10:     $S \leftarrow \min\{S, \bar{w}_{v_i} - T_{v_i}\}$
11: Return $T_{v_p} - T_{v_0}$
```

This procedure is not as fast as that of Vidal et al. (2013) which exhibits an $\mathcal{O}(1)$ complexity. Nevertheless, it seems sufficient for our tests because, in postal services, the majority of the customers do not have a time window and, therefore, waiting before servicing a customer is rare. In this case, the duration U_r of a route $r = (v_0, \ldots, v_p)$ is often well approximated by $T_{v_p} - T_{v_1} + t_{v_0,v_1}$, where T_{v_1} and T_{v_p} are the earliest service start time at v_1 and arrival time at v_p, two variables that are available in the tabu search algorithm when dealing with time windows. In fact, $U_r \leq T_{v_p} - T_{v_1} + t_{v_0,v_1}$ and, therefore, if $T_{v_p} - T_{v_1} + t_{v_0,v_1} \leq U^{max}$, there is no need to compute U_r. Finally, because the best position according to the travel time to insert a customer in a route is often the position that yields the minimal route duration when waiting is not frequent, the route duration constraint is only checked once for each customer that can be inserted in a route, namely, after finding this best insertion position.

3.2.2 Compactness cost update

Unlike the route duration which is subject to a hard constraint, route compactness is seen as a preference that incurs a cost. Therefore, this cost as to be updated, if needed, at each customer insertion and removal. Note, however, that this computation does not have to be performed for every possible insertion position of a customer because the proposed compactness measures do not depend on the sequence of the customers in a route. Below, we only discuss how this update is performed when the rectangle and the disk compactness measures are applied. Updating the compactness cost for the hybrid measure simply combines the updates for the other two measures.

Rectangle compactness measure: First, recall that, for a route r visiting the subset of customers V_r,

$$\kappa_r^R = \min_{\theta \in \Theta} \kappa_r^R(\theta),$$

where

$$\kappa_r^R(\theta) = \min\{\mu, x_r^{Max}(\theta) - x_r^{Min}(\theta)\} \times \min\{\mu, y_r^{Max}(\theta) - y_r^{Min}(\theta)\}$$

and $x_r^{Max}(\theta)$, $x_r^{Min}(\theta)$, $y_r^{Max}(\theta)$, and $y_r^{Min}(\theta)$ are the maximum and minimum θ-rotated coordinate values of the customers visited in route r. Then, observe that inserting a new customer in route r cannot decrease the values $\kappa_r^R(\theta)$, $\theta \in \Theta$, whereas removing a customer from r cannot increase these values.

Let $\Theta^* = \{\theta \in \Theta \mid \kappa_r^R(\theta) = \kappa_r^R\}$ be the subset of rotation angles in Θ yielding a rectangle of minimal area.

- When a customer $v \in V^C \setminus V_r$ with θ-rotated coordinate values (x_v^θ, y_v^θ), $\theta \in \Theta$, is inserted in route r to yield a new route r', it is easy to show that $\kappa_{r'}^R(\theta) = \kappa_r^R(\theta)$ for all angles $\theta \in \hat{\Theta}$, where $\hat{\Theta} = \{\theta \in \Theta \mid x_v^\theta \in [x_r^{Min}(\theta), x_r^{Max}(\theta)]$ and $y_v^\theta \in [y_r^{Min}(\theta), y_r^{Max}(\theta)]\}$. Therefore, two cases can occur:
  - If $\hat{\Theta} \cap \Theta^* \neq \emptyset$, then $\kappa_{r'}^R = \kappa_r^R$.
  - Otherwise, in the worst case, the values $\kappa_{r'}^R(\theta)$ must be computed for all $\theta \in \Theta \setminus \hat{\Theta}$ to determine $\kappa_{r'}^R$, requiring $\mathcal{O}(|\Theta||V_r|)$ operations.
- When a customer $v \in V_r$ with θ-rotated coordinate values (x_v^θ, y_v^θ), $\theta \in \Theta$, is removed from route r to yield a new route r', it is easy to show that $\kappa_{r'}^R(\theta) = \kappa_r^R(\theta)$ for all angles $\theta \in \tilde{\Theta}$, where $\tilde{\Theta} = \{\theta \in \Theta \mid x_v^\theta \in]x_r^{Min}(\theta), x_r^{Max}(\theta)[$ and $y_v^\theta \in]y_r^{Min}(\theta), y_r^{Max}(\theta)[\}$. Again, two cases can occur:
  - If $\tilde{\Theta} = \Theta$, then $\kappa_{r'}^R = \kappa_r^R$.
  - Otherwise, in the worst case, the values $\kappa_{r'}^R(\theta)$ must be computed for all $\theta \in \Theta \setminus \tilde{\Theta}$ to determine $\kappa_{r'}^R$, taking $\mathcal{O}(|\Theta||V_r|)$ operations.

Consequently, for the rectangle measure, the worst-case complexity of updating the compactness cost of a single route r in a tabu search iteration is given by $\mathcal{O}(|\Theta||V_r||V^C|)$, *i.e.*, $\mathcal{O}(|\Theta||V_r|)$ operations for each potentially inserted or removed customer in V^C.

Disk compactness measure: Recall that, for a route r,

$$\kappa_r^D = \left(\frac{D_r}{2}\right)^2 \pi,$$

where $D_r = \max_{(v_i,v_j)\in V_r^2} e_{v_i,v_j}$. Let $V^* = \{v_1^*, v_2^*\}$ be the set containing the two customers in V_r defining the diameter D_r, i.e., $D_r = e_{v_1^*,v_2^*}$.

- When inserting a customer $v \in V^C \setminus V_r$ in route r to yield a new route r', we compute $D_{r'} = \max\{D_r, \max_{v'\in V_r} e_{v,v'}\}$ in $\mathcal{O}(|V_r|)$ and set $\kappa_{r'}^D = \left(\frac{D_{r'}}{2}\right)^2\pi$.
- When removing a customer $v \in V_r$ from route r to yield a new route r', two cases can occur:
  - If $v \notin V^*$, then $\kappa_{r'}^D = \kappa_r^D$.
  - Otherwise, the diameter $D_{r'}$ needs to be computed from scratch as $D_{r'} = \max_{(v_i,v_j)\in V_{r'}^2} e_{v_i,v_j}$ in $\mathcal{O}(|V_r|^2)$.

Thus, in a tabu search iteration, updating the compactness cost of a single route r under the disk measure can be performed in a complexity of $\mathcal{O}(|V^C \setminus V_r||V_r|) + \mathcal{O}(|V_r|^2)$ because only the two customers v_1^* and v_2^* induce a complete calculation when a customer in V_r is checked for removal.

Comparison with the exact compactness measure: If the exact compactness measure was used to compute the compactness costs in the LNS heuristic, the compactness cost $\kappa_{r'}^E$ of a route r' (*i.e.*, the area of its convex hull) obtained from a current route r by inserting a customer $v \in V^C \setminus V_r$ or deleting a customer $v \in V_r$ could be computed as follows in the tabu search algorithm. First, we check if v is located inside the convex hull of route r using the ray casting algorithm (Shimrat, 1962), which works in $\mathcal{O}(h)$, where h is the number of edges of the hull ($h = |V_r|$ in the worst case). Then, two cases can happen:

- If customer v is inside the convex hull of route r, then $\kappa_{r'}^E = \kappa_r^E$.
- Otherwise, the convex hull of route r' must be computed and its area $\kappa_{r'}^E$ computed as explained in Section 2.1.1. In the worst case, these computations require $\mathcal{O}(|V_r|\log|V_r|)$ operations.

Consequently, updating the compactness costs resulting from the potential insertion or removal of all customers from a single route r in a tabu search iteration has a worst-case complexity of $\mathcal{O}(|V^C||V_r|\log|V_r|)$. This complexity is clearly larger than that of the disk measure. It is also larger than that of the rectangle measure when $|\Theta| < \log|V_r|$. Note, however, that with the exact measure, $\mathcal{O}(|V^C||V_r|)$ operations are mandatory to check whether the customers are inside the convex hull of the current route and determine if additional computations are necessary for each customer, whereas with the rectangle measure, $\mathcal{O}(|V^C||\Theta|)$ operations are required to assess this. Given that, for our tests, $|\Theta|$ is much less than $|V_r|$, we believe that the practical complexity of updating the compactness costs is much smaller with the rectangle measure than with the exact measure. In fact, our test results show that the rectangle measure performs either better or very similar to the disk measure.

4 Computational results on academic instances

As mentioned in the introduction, our goal is not to evaluate the performance of the proposed LNS heuristic, but rather to compare the compactness measures and analyze their impact on the solution characteristics. To a lesser extent, we also evaluate the impact of using these compactness measures on the computational time of our heuristic.

In this section, we present the computational results obtained on academic benchmark instances derived from the VRPTW instances of Gehring and Homberger (2001) which will be referred to as the GH instances. The size of these instances ranges from 200 to 1000 customers but we limited our tests to those involving 200 and 400 customers. For each size, there are three groups of 20 instances: Clustered (C), Random (R) and Random-Clustered (RC), which is a mix between the first two categories.

Because the C group is not very representative of postal services' instances and not relevant from a compactness point of view (most optimal routes are compact without incentives), our tests have focused on the two last categories (R & RC). Moreover, each group is divided according to two types of time horizons: short and long. In the short-horizon instances, a single vehicle can visit around 10 customers, which is far below the average for postal services. Therefore, we retained the instances of the groups R2 and RC2 (with about 50 customers per vehicle). The 40 selected VRPTW instances (20 with 200 customers and 20 with 400 customers) were simply converted into Comp-VRPTW by considering the compactness costs. Because it is not straightforward to assign a practical value to the maximum route duration U^{max} parameter for these instances, the maximum duration constraint has been omitted. This allows us to compare the computed Comp-VRPTW solutions and the best-known VRPTW solutions reported on the SINTEF (2019) website. The latter solutions are referred to as the *TT-BKS* to highlight that they seek to minimize the total travel time. Note that, in the GH instances, the travel times are equal to the Euclidian travel distances and that, for our tests, the travel times are truncated to the second decimal digit before being multiplied by 100.

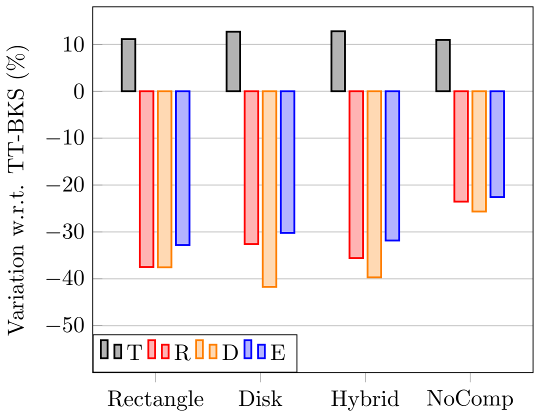
Below, we only present results obtained by varying certain parameter values impacting only the TDR phase. We exclude from these results and the discussions the VNR phase which is assumed pre-executed, *i.e.*, for each instance, the starting point of each TDR phase is the same independently of the TDR parameter setting. Unless otherwise specified, the LNS heuristic executes 50 LNS iterations in the TDR phase. It removes 70 customers (for the 200-customer instances) or 100 (for the 400-customer ones) at each iteration. When the rectangle or hybrid compactness measure is used, the set of rotation angles Θ is defined according to a discretization of the interval $[0, 90[$. By default, we chose to discretize it with an increment of $\gamma = 15$, *i.e.*, $\Theta(\gamma) = \{0, 15, 30, 45, 60, 75\}$.

4.1 Comparison of compactness measures

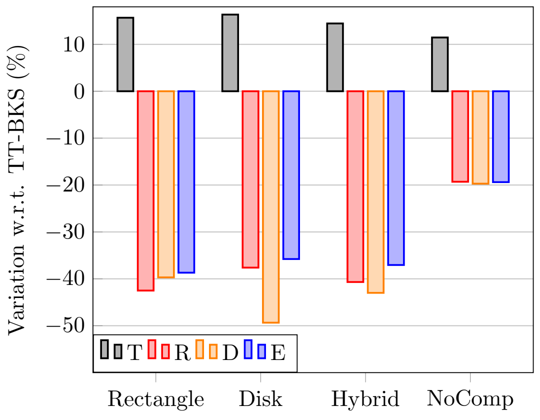
We start by comparing the compactness measures. All 40 instances were solved using four variants of the LNS heuristic which only differ by the compactness measure used: Rectangle ($\lambda^R = k, \lambda^D = 0$), Disk ($\lambda^R = 0, \lambda^D = k$), Hybrid ($\lambda^R = \lambda^D = k/2$), and NoComp ($\lambda^R = \lambda^D = 0$), where k is set to 3 in order to obtain solutions where the total compactness cost is equal to approximately 20% of the total travel cost. A sensitivity analysis on the value of λ^R when applying the Rectangle variant will be presented in Section 4.3. Note that the NoComp variant does not consider the compactness features. Each computed solution is evaluated *a posteriori* with respect to all compactness measures, including the exact measure (based on the route convex hull). For example, the solution of an instance that was computed using the rectangle compactness measure in the LNS heuristic is evaluated at the end of the optimization process using the disk, hybrid, and exact measures. Furthermore, the four compactness measures are computed for all *TT-BKS*.

The results of these tests are summarized in Figures 7a and 7b for the 200- and 400-customer instances, respectively. For each algorithm variant (identified on the horizontal axis by the compactness measure used), the four vertical bars indicate the average variation in percentage of the travel (routing) cost (T, black), the rectangle (R, red), disk (D, orange), and exact (E, blue) compactness costs of the computed solution with respect to the *TT-BKS*. For example, the set of four bars above Disk in Figure 7a show that, compared to the *TT-BKS*, the travel cost of the solution computed by the Disk variant of the LNS algorithm is on average 12.7% larger but that the rectangle, disk, and exact compactness costs reduce by 32.6%, 41.7%, and 30.2%, respectively. Note that the travel cost excludes any vehicle fixed cost and that for some 400-customer instances, our LNS algorithm was unable to reach in the VNR phase the minimum number of vehicles used in the *TT-BKS*.

For both instance sets, the improvement in the compactness costs is significant, varying from 32.6% to 49.4% depending on the instances and the measure used in the algorithm. The final improvement of the exact measure value is also quite steady, ranging from 30.2% to 38.7%. The tradeoff is achieved by increasing the travel cost by a substantial amount (between 11.1% and 16.3%). As the NoComp results show, our method (especially when starting from an initial solution favoring compactness and using such a low number of LNS iterations) cannot compete with state-of-the-art algorithms to minimize the travel time only. However, when dealing with compactness, the tradeoff is unavoidable as it will be further discussed in Section 4.3. Comparing the compactness results of the first three algorithm variants to the NoComp variant, we observe that much larger improvements are obtained when compactness is considered in the algorithm.



(a) 200 customers



(b) 400 customers

Figure 7: Comparison of compactness measures

Concerning the compactness measures used in the LNS heuristic, we do not observe a large difference on the average variation of the exact measure value (between 30.2% and 32.8% for the 200-customer instances and between 35.8% and 38.7% for the 400-customer instances) they induce. Nevertheless, the rectangle measure yields the largest average improvement for both instance sets, whereas the disk measure is the less effective although it succeeds to reduce the disk measure value the most. To further compare the compactness measures, Table 2 reports the average computational time (CPU) required in the TDR phase by each algorithm on each set of instances as well as the average time increase (Δ CPU) with respect to the time required by the NoComp algorithm variant. These results show that dealing with compactness can moderately increase the average computational times of the LNS heuristic. This increase is larger in proportion for the 200-customer instances because solving the restricted master problems requires much more time for the 400-customer instances. Furthermore, these results indicate that the rectangle compactness measure is less time-consuming than the disk measure and the hybrid measure which combines the other two. Consequently, it appears that the rectangle measure offers the best performance in terms of both route compactness quality and computational time.

Table 2: Average computational times for the GH instances

	200 customers		400 customers	
	CPU (s)	Δ CPU (%)	CPU (s)	Δ CPU (%)
NoComp	127	–	455	–
Rectangle	188	48.0	517	13.6
Disk	191	50.4	588	29.2
Hybrid	231	81.9	665	46.2

4.2 Sensitivity to rotation angle discretization

To evaluate the sensitivity of the rectangle measure to the angle discretization increment γ as defined in the previous section, we ran tests with the Rectangle variant ($\lambda^R = 3, \lambda^D = 0$) of the LNS heuristic using different γ values, namely, $\gamma = 2, 5, 15, 30, 45, 90$. For each instance and each γ value, we obtained a solution which was evaluated *a posteriori* according to three different compactness measures: $\sum_r K_r^R(\Theta(\gamma))$, $\sum_r K_r^R(\Theta(1))$, and $\sum_r \kappa_r^E$, where the sums are taken over the routes in the solution and $K_r^R(\Psi) = \min_{\psi\in\Psi} \kappa_r(\psi)$. The first measure is the rectangle measure used by the algorithm, i.e., with γ to determine the rotation angle set Θ. The second is the rectangle measure with a discretization increment $\gamma = 1$ which allows to estimate the error committed by a coarser discretization. Finally, the last one is the exact compactness measure which allows to determine the error induced by replacing the area of the route convex hull by the area of the best rotated rectangle.

The results of these experiments are presented in Figure 8 which is divided in two parts according to the size of the instances. The red, green and blue curves indicate the average compactness costs in function of γ for the above three measures, respectively. The black curve illustrates the average increase in the TDR phase computational time with respect to the time obtained for the no rotation case (γ = 90). Rationally, one can expect that the lower the value of γ, the smaller the difference between $\sum_r K_r^R(\Theta(\gamma))$ and $\sum_r K_r^R(\Theta(1))$, and the longer it takes to execute the 50 LNS iterations in the TDR phase. In practice, these tendencies are observed but not with a perfect monotonicity. On the other hand, the impact of decreasing the value of γ on the exact measure value $\sum_r \kappa_r^E$ is not significant on average.

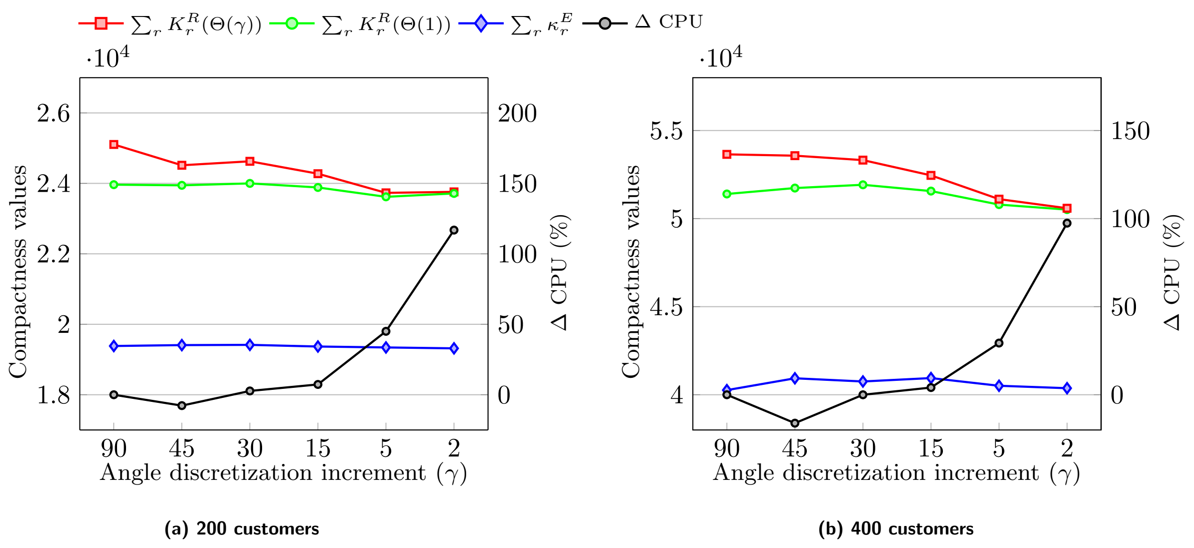


Figure 8: Sensitivity of the compactness costs and computational time to the value of γ

To pursue our analysis, we extracted additional statistics from the solutions obtained with different γ values, but also the solutions produced by the Disk variant and the Hybrid variant (with $\gamma = 15$) of the LNS heuristic. More precisely, for each algorithm, instance and route in the solution obtained, we computed the relative error between the compactness costs evaluated according to the measure used in the algorithm and the exact measure. For each instance size, Table 3 reports the average of these relative errors (Err.) and the standard deviation (σ). With regards to the rectangle measure, we observe a substantial gain when considering rotated rectangles, even with a single 45°-rotation ($\gamma = 45$). Taking into account the Δ CPU values of Figure 8, a value of $\gamma = 15$ seems to be a good tradeoff between measure accuracy and computational time. For the disk and hybrid measures, the errors are much larger than with the rectangle measure. In fact, we have noticed that, in general, the disks do not fit the routes as well as the rectangles. In addition, we have observed some pathological routes that are much better approximated by a rectangle with a large length compared to its width than by a disk.

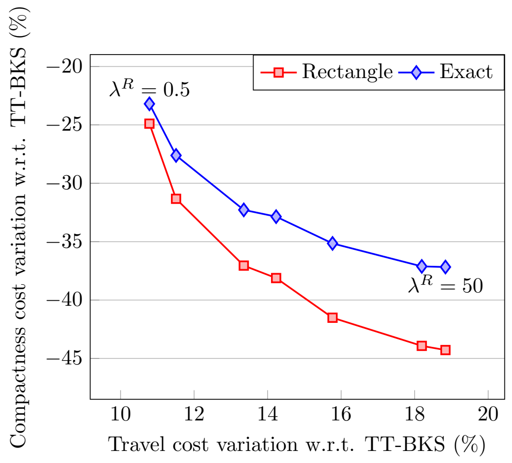
4.3 Sensitivity to rectangle measure weight

In this section, we evaluate the sensitivity of the value of the weight parameter λ^R that is used to put more or less emphasis on reducing the compactness cost with respect to the travel cost when the Rectangle variant ($\lambda^R > 0$, $\lambda^D = 0$) of the LNS heuristic is applied. To do so, we solved all 40 GH instances using different λ^R values, *i.e.*, $\lambda^R \in \{0.5, 1, 3, 5, 10, 20, 50\}$. The plots in Figure 9, for 200- and 400-customer instances, show the average variations (with respect to the TT-BKS) of the compactness costs and the travel (routing) costs in function of the weight λ^R, where its value increases from left to right. The blue curve is associated with the exact compactness measure whose value is computed *a*

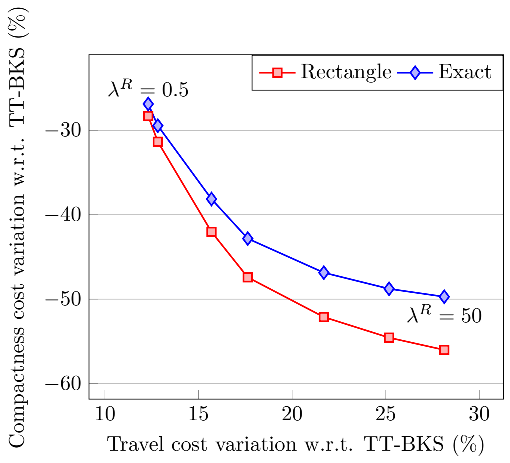
Table 3: Relative error with respect to the exact measure and its standard deviation

		200 customers		400 customers	
Measure	γ	Err. (%)	σ (%)	Err. (%)	σ (%)
Rectangle	90	28.9	20.3	34.9	20.5
	45	25.8	10.5	31.6	13.2
	30	26.3	10.8	31.9	12.4
	15	23.8	9.1	29.5	11.0
	5	22.8	9.6	26.5	10.4
	2	21.0	8.5	26.1	11.3
Disk		49.7	19.7	50.1	31.1
Hybrid	15	41.6	16.9	52.0	25.8

posteriori, whereas the red curve refers to the rectangle measure, with $\gamma = 15°$. Each curve defines an approximate Pareto-front. The larger the λ^R value, the better the improvement in the compactness cost, but the larger the travel cost. As a reference to evaluate the incurred tradeoff, when $\lambda^R = 1$, the compactness cost is worth about 5% to 10% of the travel cost.



(a) 200 customers



(b) 400 customers

Figure 9: Sensitivity of the compactness and travel costs to the value of λ^R

4.4 Impact of initial solution

In this section, we discuss the impact of using the initial solution computed by the approach presented in Section 3.1 and referred to as IS below. In fact, we compare its usage with using the TT-BKS as the initial solution. Starting from the TT-BKS, all 40 GH instances were, thus, solved again using the Rectangle variant of the LNS heuristic and the different λ^R values specified in Section 4.3. Figure 10 provides two tradeoff curves between the average variations (with respect to the TT-BKS) of the compactness cost (computed using the exact measure) and the travel cost in function of λ^R. This time, the averages are computed over the 200- and 400-customer instances together. The blue curve is obtained when starting the LNS heuristic from IS (therefore, it corresponds to the average of the two blue curves in Figure 9), whereas the gray one arises when it begins from TT-BKS.

Recall that the procedure used to compute IS favors compactness. This is confirmed by Figure 10, where one can see that starting from IS enables to find very compact final solutions even with a small weight λ^R. With larger weights, these starting points generate the best solutions relatively to the compactness criterion in the limited number of LNS iterations performed. However, if a large travel cost increase cannot be afforded to improve compactness, it seems better to start from an initial solution that has a travel cost close to that of the TT-BKS.

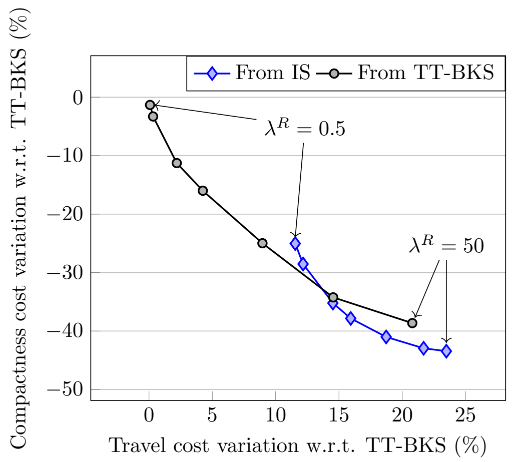


Figure 10: Initial solution impact with respect to varying λ^R values

4.5 Examples of compact and non-compact solutions

In this section, using solutions of the RC2_2_3 instance with 200 customers, we show how compact and non-compact solutions can look like, and how the rectangle and disk compactness measures can be interpreted visually. In Figures 11a and 11b, we present the TT-BKS and the solution obtained by the Rectangle variant of the LNS heuristic, respectively (the black square represents the depot). Both solutions comprise four routes that are highlighted with different colors. For each route, the corresponding rectangle is the one used to compute its compactness and the shaded polygon corresponds to its convex hull. For the TT-BKS, we obtain the following statistics:

$$\sum_{r\in\Omega^B} \tau_r = 260186, \quad \sum_{r\in\Omega^B} \kappa_r^E = 27025, \quad \sum_{r\in\Omega^B} \kappa_r^R = 35881, \quad \min_{r\in\Omega^B} \kappa_r^R = 4824, \quad \max_{r\in\Omega^B} \kappa_r^R = 17015,$$

where Ω^B is the set of routes in the TT-BKS. For the Rectangle variant heuristic solution composed of the set of routes Ω^R, we get:

$$\sum_{r\in\Omega^R} \tau_r = 290990, \quad \sum_{r\in\Omega^B} \kappa_r^E = 16506, \quad \sum_{r\in\Omega^R} \kappa_r^R = 20379, \quad \min_{r\in\Omega^R} \kappa_r^R = 3520, \quad \max_{r\in\Omega^R} \kappa_r^R = 7308.$$

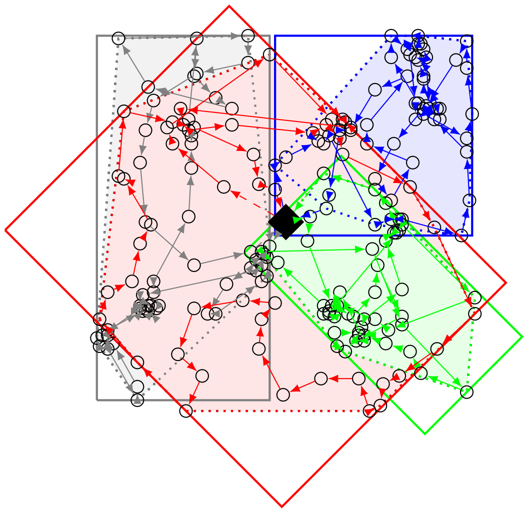
Observe first that, compared to the TT-BKS, the travel cost of the Rectangle variant solution increases by close to 12% but the total exact (resp., rectangle) compactness cost decreases by 43% (resp., 39%). In both solutions, the two routes on the right (blue and green) cover each approximately the same area in both solutions, but the other two routes differ significantly from one solution to the other. In fact, one can see that the red route in the TT-BKS travels almost all around the whole region (except the top-right corner) and induces a large compactness cost. Observe also that route overlapping is very limited in the compact Rectangle variant solution even if this criterion is not taken into account in the objective function.

Similar to Figures 11a and 11b, Figures 12a and 12b allow to compare the TT-BKS with the solution computed by the Disk variant LNS heuristic, using the disk measure to evaluate compactness. The TT-BKS exhibits the following statistics:

$$\sum_{r\in\Omega^B} \tau_r = 260186, \quad \sum_{r\in\Omega^B} \kappa_r^E = 27025, \quad \sum_{r\in\Omega^B} \kappa_r^D = 46306, \quad \min_{r\in\Omega^B} \kappa_r^D = 5621, \quad \max_{r\in\Omega^B} \kappa_r^D = 17691,$$

whereas the Disk variant heuristic solution, with a set of routes Ω^D, has the following ones:

$$\sum_{r\in\Omega^D} \tau_r = 286372, \quad \sum_{r\in\Omega^B} \kappa_r^E = 16457, \quad \sum_{r\in\Omega^D} \kappa_r^D = 24387, \quad \min_{r\in\Omega^D} \kappa_r^D = 4092, \quad \max_{r\in\Omega^D} \kappa_r^D = 7852.$$


(a) TT-BKS

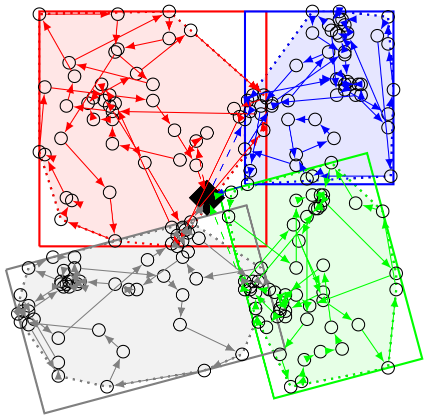
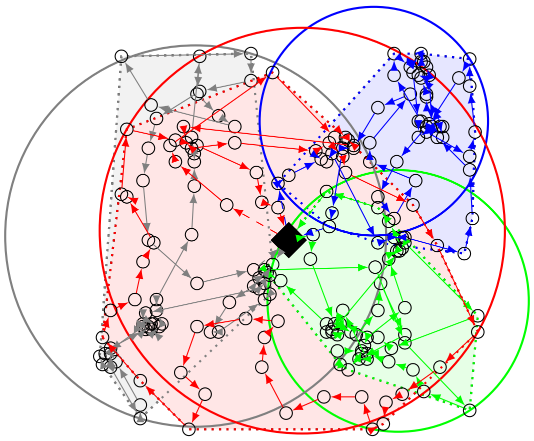

(b) Rectangle variant solution

Figure 11: Structure of the TT-BKS and the Rectangle variant solution


(a) TT-BKS

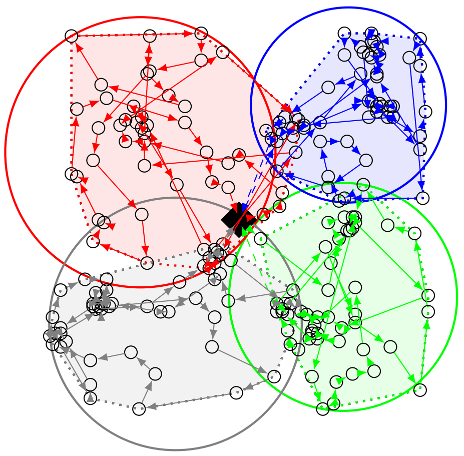

(b) Disk variant solution

Figure 12: Structure of the TT-BKS and the Disk variant solution

Here, we observe a travel cost increase of approximately 10% to obtain a large 39% decrease of the total exact compactness measure (47% of the disk measure). This decrease is more substantial than the one reported above for the rectangle measure. This is due to the fact that, for this example, the disk measure is less accurate than the rectangle measure and greatly overestimates the area of the convex hull for certain routes of the TT-BKS, especially for the leftmost route (in gray). This higher inaccuracy induces a large difference between the compactness costs (areas of rectangles or disks) of the Rectangle and Disk variant solutions (20379 and 24387) although these solutions are quite similar.

5 Computational results on instances derived from a real-world dataset

Our industrial partner Giro Inc., which commercializes optimization software to postal agencies, gave us one real-world Comp-VRPTW instance with 1553 customers, including only 16% with a time window. Beside the customer time windows, the data specify the estimated travel time between each pair of locations, the service time at each customer, and the geographical coordinates of each customer location, allowing us to compute the Euclidian distance between each pair of customers. From this instance, we extracted subsets of customers randomly to create a total of 12 instances, namely, three instances of each of the following sizes: 200, 300, 400 and 500 customers. Our computational experiments have focussed on these smaller-sized instances as our LNS heuristic is not well suited to solve the large original instance. Tackling it would require the development of acceleration strategies and is, thus, left for future work.

These 12 instances are hereafter called the industry-derived instances and some of their characteristics are provided in Table 4. For each instance size (# Customers) and each of the three instances of this size (denoted $I1$, $I2$ and $I3$), this table indicates the number of customers with a time window (# TWs) and the number of vehicles (# Vehicles) used in the solution obtained at the end of the VNR phase of the LNS heuristic. One can observe that the percentage of customers with a time window varies between 12% and 19.5% and that the average number of customers per vehicle ranges between 60 and 80. For all these instances, the maximum route duration U^{max} is set to 8 hours.

Table 4: Characteristics of the industry-derived instances

	# TWs			# Vehicles		
# Customers	$I1$	$I2$	$I3$	$I1$	$I2$	$I3$
200	29	26	39	3	3	3
300	36	46	58	4	4	5
400	63	54	55	6	5	5
500	82	78	81	7	7	7

Each industry-derived Comp-VRPTW instance was solved four times using the four variants of the proposed branch-and-price-based LNS heuristic: Rectangle ($\lambda^R = 2k$, $\lambda^D = 0$), Disk ($\lambda^R = 0$, $\lambda^D = k$), Hybrid ($\lambda^R = k$, $\lambda^D = 0.5k$), and NoComp (without considering compactness, i.e., $\lambda^R = \lambda^D = 0$), where k is a predefined constant such that the total compactness cost of a compact solution represents about 20% of its total travel cost. Note that less weight is put on the disk measure compared to the rectangle measure because the former tends to overestimate more the exact measure as shown by the results in Table 3. When the rectangle or hybrid compactness measure is used, the rotated angle discretization increment is set to $\gamma = 15$. In all cases, the initial solutions are computed as explained in Section 3.1. For instances with 200 customers, the set of customers is divided into 4 subsets, whereas, for the other instances, 9 subsets are used. As in the previous section, the VNR phase is the same for all heuristic variants, while the TDR phase differs according to the compactness measure applied. For all variants, 100 LNS iterations are performed in the TDR phase.

In the following, we report computational results where the compactness is defined using geographical areas (Section 5.1) and using maximum travel times (Section 5.2).

5.1 Compactness based on geographical areas

As in Section 4.1, we compare the routing/compactness cost tradeoff induced by the solutions obtained with each algorithm variant taking into account compactness when measured using geographical areas, as before. The results for the industry-derived instances are summarized in Figure 13 which is crafted as a subfigure of Figure 7, except that the variations are computed with respect to the NoComp variant solutions (instead of the TT-BKS).

From these results, we observe again that taking into account compactness during optimization (Rectangle, Disk and Hybrid variants) can yield much more compact routes, with an average gain of the exact measure varying between 9.7% and 11.3%. The travel cost results show that minimizing compactness helps to further diminish the travel costs (average reductions varying between 0.2% and 1.1%). This can be explained by the fact that, when the percentage of customers with a time window is small as in the industry-derived instances, routes minimizing the total travel cost naturally tend to be compact and, consequently, favoring compactness also favors shorter routes. Comparing the performance of the three heuristic variants comnsidering compactness, one can see that the Disk variant yields slightly better solutions with respect to the exact compactness measure and the travel costs, even though the disk measure does not seem as reliable as the rectangle and hybrid measures.

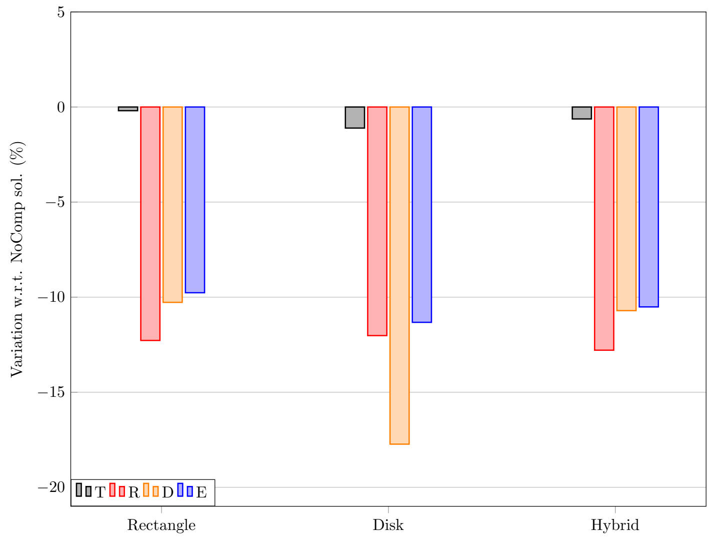


Figure 13: Comparative results for industry-derived instances

In Table 5, we report the average computational time (CPU) required in the TDR phase by each algorithm variant as well as the average time increase (Δ CPU) compared to the reported NoComp time. These results indicate again that handling the compactness costs increases the computational times of our LNS heuristic moderately (between 25.5% and 34.2% on average). This increase is, however, less important (especially for the Hybrid variant) than for the academic instances (see Table 2), probably because the objectives of minimizing compactness costs and of minimizing travel costs are less conflicting when there are less time windows as mentioned above.

Table 5: Average computational times for the industry-derived instances

Variant	CPU (s)	Δ CPU (%)
NoComp	509.2	–
Rectangle	664.7	30.6
Disk	643.9	26.5
Hybrid	683.2	34.2

5.2 Compactness based on maximum travel times

We mentioned in Section 2.1.3 that other measures than areas based on Euclidean distances could be used to evaluate the compactness of a route. Using travel time is particularly useful when Euclidean distances do not accurately reflect the road network. In this section, we propose to determine if the disk measure where the diameter D_r of a route is defined by the maximum travel time between two customers in this route can help reducing the geographical compactness cost of the computed routes. Hereafter, the heuristic variant using this travel time disk measure is called the Disk-TT variant.

All industry-derived instances where solved using the Disk-TT variant. A posteriori, we computed the geographical exact compactness measure of each route in each computed solution. Furthermore, the solutions produced by the Rectangle, Disk and NoComp variants were also evaluated a posteriori with respect to the travel time disk compactness measure. Figure 14 allows to compare the Rectangle, Disk and Disk-TT variant solutions with respect to the travel costs (T), the travel time disk compactness costs (DTT), and the geographical exact compactness costs (E). Again, the bars indicate the average variations of these costs with respect to the costs incurred by the NoComp variant solutions. The results show that the Disk and Disk-TT variants perform similarly with respect to the travel time disk measure. However, on average, the Disk-TT variant solutions have smaller total travel costs but larger geographical exact compactness costs. This is not surprising because minimizing the travel time disk measure can avoid assigning to the same route pairs of customers that are geographically close but not easily linkable by the road network. As for the Rectangle variant, it generates solutions that are not competitive with the other variants when considering the travel time disk measure because this variant does not necessarily disadvantage routes that cover areas which are longer than wider, inducing a larger maximum travel time disk measure. Finally, for the Disk-TT variant, we report an average computational time of 787.9 seconds, which is an increase of 54.7% compared to the NoComp variant. This increase is somewhat larger than those observed for the other variants (see Table 5).

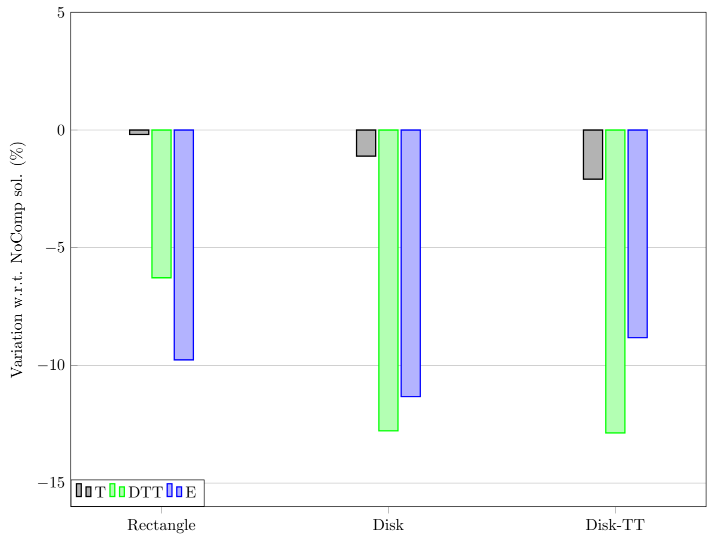


Figure 14: Comparative results involving the Disk-TT variant

6 Conclusion

In this paper, we proposed two compactness measures to favor generating compact routes when solving the Comp-VRPTW. Considering the exact compactness measure of a route as the area of its convex hull, we approximated this measure by replacing the convex hull by either a rectangle or a disk. When geographical measures based on Euclidean distances are not suitable due to the actual road network, the disk approximation can be adapted by redefining the disk diameter as, e.g., the maximum travel time between two customers in the route. Next, to take into account the minimization of these compactness measures, including a hybridization of the rectangle and disk measures, we modified the branch-and-price-based LNS heuristic of Prescott-Gagnon et al. (2009). Our computational experiments on academic Comp-VRPTW instances with time windows for all customers showed that the proposed heuristic variants can greatly reduce the compactness costs by around 35% in exchange of increasing the travel costs by more than 10%. On the other hand, for industry-derived instances involving up to 500 customers with less than 20% subject to a delivery time window, favoring compactness during the optimization achieved a gain of about 10% on the compactness costs while also slightly improving the travel costs. In all cases, handling compactness increases the average computational times by 13% to 81%. Finally, we have seen that when using travel time instead of Euclidean distance to define the diameter in the disk measure, it provides some improvement in maximum-time-travel compactness but also in geographical compactness.

As future work, we want to develop speedup strategies for the proposed LNS heuristic for efficiently tackling real-life instances containing several thousands customers.

A Constraint programming model used to compute an initial solution

To compute an initial solution for the proposed branch-and-price-based LNS heuristic, we use the CPLEX constraint programming optimizer (CP-Optimizer, for additional details see IBM ILOG CPLEX, 2017) and apply it on the following model which we describe using the CP-Optimizer variables and constraints. First, for each customer $i \in V^C$, we define a master task as a mandatory interval variable A_i and its optional copies B_i^l for each allowed vehicle $l \in L$. For each vehicle l, we also add extra interval variables $\underline{D}^l$ and $\overline{D}^l$ representing the depot at the start and the end of its route. Earliest start time, latest start time and length are predefined attributes of interval variables, which are used to model the time window and service time (if any) at each customer. A route assigned to vehicle l is then defined by a SequenceVariable Seq^l involving $\left(\overline{D}^l \cup (\cup_{i \in V^C} B_i^l) \cup \underline{D}^l\right)$. Denoting by T_X the start time of a task X and by $[\![\chi]\!]$ the boolean value of expression χ, the constraint programming model is as follows:

$$\min \quad (\sum_{l \in L} T_{\overline{D}^l} - T_{\underline{D}^l}) + F \times \sum_{l \in L} [\![T_{\overline{D}^l} - T_{\underline{D}^l} > 0]\!] \tag{10}$$

$$\text{subject to:} \qquad Seq^l.NoOverlap(\Upsilon), \qquad \forall l \in L \tag{11}$$

$$Seq^l.First(\underline{D}^l), \qquad \forall l \in L \tag{12}$$

$$Seq^l.Last(\overline{D}^l), \qquad \forall l \in L \tag{13}$$

$$T_{\overline{D}^l} - T_{\underline{D}^l} \leq U^{max}, \qquad \forall l \in L \tag{14}$$

$$Alternative(A_i, \cup_{l \in L} B_i^l), \qquad \forall i \in V^C. \tag{15}$$

Objective function (10) seeks to minimize the sum of the route durations and the fixed cost for each vehicle used. Indeed, $T_{\overline{D}^l} - T_{\underline{D}^l} > 0$ when at least one customer is visited in the route selected for vehicle l. Given the transition distance matrix Υ, the $NoOverlap$ constraints (11) ensure that there is sufficient time to travel between two customers visited consecutively in a route. To properly define the sequences, constraints (12) and (13) impose that each route starts and ends at the depot, respectively. The partitioning of the customers $i \in V^c$ in the different vehicles l is handled by the *Alternative* constraints (15) on A_i and B_i^l. These constraints work as follows: if A_i is present (which is mandatory in our case), then exactly one variable in $\cup_{l \in L} B_i^l$ must also be present. This forces each customer to be visited once and only once by a single vehicle.

References

N. Agatz, P. Bouman, and M. Schmidt. Optimization approaches for the traveling salesman problem with drone. Transportation Science, 52(4):965–981, 2018.

F. Arnold and K. Sörensen. What makes a VRP solution good? The generation of problem-specific knowledge for heuristics. Computers & Operations Research, 106:280–288, 2019.

T. M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. Discrete & Computational Geometry, 16(4):361–368, 1996.

L. Costa, C. Contardo, and G. Desaulniers. Exact branch-price-and-cut algorithms for vehicle routing. Transportation Science, 53(4):946–985, 2019.

G. B. Dantzig and J. H. Ramser. The truck dispatching problem. Management Science, 6(1):80–91, 1959.

G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transportation Science, 42(3):387–404, 2008.

G. Desaulniers, O. B. G. Madsen, and S. Ropke. The vehicle routing problem with time windows. In P. Toth and D. Vigo, editors, Vehicle Routing: Problems, Methods and Applications, chapter 5, pages 119–159. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2nd edition, 2014.

H. Gehring and J. Homberger. A parallel two-phase metaheuristic for routing problems with time-windows. Asia Pacific Journal of Operational Research, 18(1):35–48, 2001.

R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. Information Processing Letters, 1:132–133, 1972.

IBM ILOG CPLEX. Optimization studio CP optimizer user's manual, version 12.8, 2017. URL www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/usrcpoptimizer.pdf.

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

C. C. Murray and R. Raj. The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. Transportation Research Part C: Emerging Technologies, 110:368–398, 2020.

Y. Nagata, O. Bräysy, and W. Dullaert. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Computers & Operations Research, 37(4):724–737, 2010.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. Computers & Operations Research, 34(8):2403–2435, 2007.

E. Prescott-Gagnon, G. Desaulniers, and L.-M. Rousseau. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. Networks, 54(4):190–204, 2009.

R. Z. Ríos-Mercado and E. Fernández. A reactive GRASP for a commercial territory design problem with multiple balancing requirements. Computers & Operations Research, 36(3):755–776, 2009.

M. Shimrat. Algorithm 112: Position of point relative to polygon. Communications of the ACM, 5(8):434, 1962.

SINTEF. Gehring and Homberger benchmark, 2019. URL www.sintef.no/projectweb/top/vrptw/homberger-benchmark. Accessed: 2019-10-31.

T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Computers & Operations Research, 40(1):475–489, 2013.