Small and Large Neighborhood Search for the Park-and-Loop Routing Problem with Parking Selection

Théo Le Colleter, Dorian Dumez, Fabien Lehuédé, Olivier Péton

Small and Large Neighborhood Search for the Park-and-Loop Routing Problem with Parking Selection

Théo Le Colleter [a], Dorian Dumez [a,b], Fabien Lehuédé [a], Olivier Péton [a]
[a] IMT Atlantique, LS2N, UMR CNRS 6004, Nantes, France
[b] Department of Logistics and Operations Management, HEC Montréal, H3T 2A7, Canada
lecolleter.theo@gmail.com, {fabien.lehuede;olivier.peton}@imt-atlantique.fr, dorian.dumez@hec.ca

Abstract

This paper presents a variant of the vehicle routing problem regarding the delivery of products to customers in cities with a combination of walking and driving. The objective is first to offer a better modeling of delivery problems in congested cities and also to evaluate potential savings in traveled distances and parking times. We introduce the Park-and-Loop Routing Problem with Parking Selection (PLRP-PS) in which a parking space or loading zone has to be found for the driver and his vehicle before he walks to deliver to one or several customers. In this paper, we will focus on cases where parking locations should be selected among a large set of parking areas. To solve this problem, we developed a variant of the large neighborhood search metaheuristic called Small and Large Neighborhood Search (SLNS). We will focus on designing and comparing simple and efficient techniques to select parking spots for vehicles before goods are delivered by walking trips. The efficiency of the approach is demonstrated in the park-and-loop routing problem, with eleven new best solutions found on an existing benchmark. Some realistic instances were generated based on open data from the city of Nantes, France. In these instances, we find that combining walking and driving to deliver to the center of a city can save 19% of working time on average compared to the classical vehicle routing approach.

Keywords: vehicle routing, park-and-loop, large neighborhood search, last-mile delivery

1. Introduction

Reducing congestion and CO_2 emissions is a major challenge for the delivery of goods in urban areas. In this paper, we are interested in combining driving and walking for last-mile delivery, with a focus on where a vehicle can park before its driver can deliver to his or her customers.

The foremost inspiration of this work is that most vehicle routing problems implicitly assume that vehicles park in front of customer locations. This assumption is not always realistic in dense cities, since parking spaces do not always exist at a customer location. Indeed, parking a delivery vehicle exactly in front of each customer is often impossible, but it may not even be profitable considering the driving distance or workload. It can be more efficient to cross the street by foot than to turn around or make a detour to park on the customer's side of the street. If two successive customers are located near each other, re-starting the vehicle and parking it a few meters further is also not realistic when the driver can walk between the two addresses. As many cities tend to reallocate road space to pedestrians, bicycles, and public transportation (Martinez-Sykora et al., 2020) it seems more and more valuable to optimize delivery routes by considering parking locations and walking trips.

Supporting this statement, recent studies show that the time to find parking spots is not negligible in large cities (Reed et al., 2021). Thus, the unproductive driving time spent looking

for a parking space leads to significant overhead costs (Bates et al., 2017). Consequently, drivers often illegally double-park, also at high costs (Chiara et al., 2020). In both cases, assuming parking at customers has a cost, it also generates congestion, air pollution and noise pollution.

To address this challenge we investigate vehicle routing with driving, walking, and parking selection for last mile delivery. In this study, the potential parking locations are the available commercial loading zones defined by local authorities. In a dense urban area, we show that it is beneficial to serve multiple customers from one parking spot.

To better integrate these considerations in VRP models and algorithms, we introduce the Park-and-Loop Routing Problem with Parking Selection (PLRP-PS). This problem considers a set of customers who are delivered to from a depot by drivers with a homogeneous fleet of vehicles. We presume a large set of parking locations is known and that drivers have to park at one of these locations before delivering to some customers through *walking-trips*.

In this paper, we first review the related literature in Section 2. Then, a formal definition and a Mixed Integer Linear Programming (MILP) formulation for the PLRP-PS are proposed in Section 3. To solve this problem, we designed a variant of Large Neighborhood Search (LNS) called Small and Large Neighborhood Search (SLNS) (Section 4). We detail the used operators, types of insertions, and we focus more specifically on efficient parking selection strategies. The computational experiments are presented in Section 5. We first evaluate the proposed algorithmic components in order to identify the best LNS configuration to solve the PLRP-PS. Then, the performance of the designed SLNS is evaluated on a particular case of the VRP-TR (Coindreau et al., 2019). Finally, managerial insights on the impact of parking selection are presented.

2. Literature review

Section 2.1 discusses VRP variants that integrate the combination of driving and walking. Section 2.2 presents some other categories of problems related to the PLRP-PS. As the main focus of this paper is the selection of parking in a commercial loading zone from a large set of possible locations, the literature review gives particular attention to this aspect.

2.1. Vehicle routing problems with park-and-loop

The combination of walking and driving in routing optimization was first introduced in a Location Arc Routing Problem by Levy and Bodin (1989), within the context of mail delivery. Later on, *park-and-loop* was described in Bodin and Levy (2000) as a practice of parking the vehicle before serving a set of nearby customers on foot before returning to the vehicle and continuing the route. This practice is also mentioned in Irnich (2008) as an extension of the windy rural postman problem. To our knowledge, the first solution method for a VRP with park-and-loop was proposed by Gussmagg-Pfliegl et al. (2011) also presented as a mail delivery problem. This method follows a cluster-first route-second heuristic combined with the solving of a set partitioning problem.

Park-and-loop is considered in the *Vehicle Routing Problem with Time Windows and Multiple Service Workers* (VRPTWMS, Senarclens de Grancy and Reimann (2015)). The VRPTWMS is an extension of the VRPTW where multiple service workers can be in the same vehicle and vehicles can park only at pre-identified parking locations. After a vehicle is parked, workers perform back-and-forth walking trips to customers from the vehicle, serving only one customer per walking trip. The vehicle may leave the parking spot when all workers have served their customers. A fast cluster-first route-second heuristic was developed. First, a parallel insertion heuristic is used to group customers and assign them to a parking location. Second, the I1

insertion heuristic of Solomon (1987) is used to determine the routes. Randomly generated instances with up to 200 customers and 150 parking locations are considered.

Martinez-Sykora et al. (2020) solve an asymmetric TSP with park-and-loop where the objective is a weighted sum of the total driving time and the total walking time. A branch-and-cut algorithm is used to solve instances from a London case study with up to 30 customers. In this problem, the vehicle is parked at a customer's location but the driver can walk to serve several customers from one parking spot.

Reed et al. (2021) describe the Capacitated Delivery Problem with Parking (CDPP). The CDPP is a VRP in which delivery persons can drive and walk. They park near the customers and can serve a few others by walking but finding a parking spot on a commercial loading zone takes a given amount of time. A Mixed Integer Programming (MIP) model is developed, with dedicated valid inequalities, to solve instances based in the State of Illinois. Extensive experiments study the impact of parking time. They conclude that taking it into account is of major importance in cities and that combining driving and walking is very promising.

Coindreau et al. (2019) introduce the *Vehicle Routing Problem with Transportable Resources* (VRP-TR). The VRP-TR considers a setting with drivers and additional workers that can carpool, walk between customers, and possibly change vehicles. The objective in this problem is to minimize the sum of all vehicles' travel distance, taking account a maximum overall walking distance for each worker as well as a day duration constraint. A Variable Neighborhood Search (VNS) metaheuristic is developed to solve instances inspired by a European energy provider with up to 50 customers with a time budget of 10 hours. These VRP-TR instances are solved both in the case where car-pooling is allowed and also with one driver per vehicle and no car-pooling. The VRP-TR also takes the set of customers as the set of parking locations.

Cabrera et al. (2022) introduce the *Doubly Open Park-and-Loop Routing Problem* (DOPLRP) which is closely related to the VRP-TR especially when carpooling is not allowed. The DOPLRP consists of designing routes for technicians to visit a set of customers for an energy provider. Routes are subcontracted and their costs are evaluated based on the distance traveled between the first and the last customer of each route. There are three types of routes: pure vehicle routes where the technician serves customers by driving only, pure walking routes where the technician serves customers by walking only, and vehicle routes with park-and-loop. Vehicles can park at customer locations. The authors propose a matheuristic which is composed of a split procedure on a giant-tour that provides routes and a set partitioning formulation that uses these routes to improve solutions. The algorithm is shown to outperform the VNS of Coindreau et al. (2019) on the VRP-TR instances in the case where carpooling is not considered. In addition, the proposed method is used to solve real-world DOPLRP instances with up to 3800 customers.

A first observation from the recent literature is that, independent of parking selection, the terminology is not unified over the different papers that integrate the VRP and park-and-loop. In the following, we denote by the Park-and-Loop Routing Problem (PLRP) that the simplest integration of park-and-loop in the VRP, whereby vehicles can park at customers and drivers can perform walking trips to deliver more than one customer from a parking location. This corresponds to the VRP-TR without car-pooling of Coindreau et al. (2019) or the DOPLRP with closed vehicle routes from Cabrera et al. (2022).

2.2. Related classes of problems

The *Truck and Trailer Routing Problem* (TTRP) was first studied by Semet (1995) under the name Partial Accessibility Vehicle Routing Problem. This problem typically occurs in milk collection problems where some producers cannot be accessed by a truck with its trailer. Thus,

the truck can leave its trailer at some customer's location and perform some visits before re-coupling with its trailer. Hence, in this problem, solutions have a park-and-loop structure. The connection between the TTRP and park-and-loop routing problems is extensively discussed in the recent paper of Cabrera et al. (2022). In particular, the TTRP with time windows has been recently addressed by Parragh and Cordeau (2017) with a branch-and-price algorithm and an LNS. The authors solve instances with up to 100 customers in which the decoupling points are the customers that are accessible with trailers.

The routing problems with drones also consider sub-tours for visiting customers. The *Traveling Salesman Problem with Drone* (Agatz et al., 2018, TSP-D) extends the standard TSP by allowing the driver to send an unmanned aerial vehicle (UAV) to deliver a package while he/she is serving other customers. The truck and the drone can couple/decouple only at customers' locations. The authors propose a mathematical model and a route-first, cluster-second method to solve instances with up to 100 customers. More recently, Li et al. (2021) defined the *Two-Echelon Vehicle Routing Problem with Time Windows and Mobile Satellites* (2E-VRP-TM). This problem considers a homogeneous fleet of vans paired with UAVs for parcel delivery. It extends the standard VRP by allowing UAVs to deliver customers while the van is parked at another customer location. An Adaptive Large Neighborhood Search (ALNS) heuristic is proposed to solve instances with up to 100 customers.

In addition, the combination of walking and driving can be found in home health care logistics. For example, Fikar and Hirsch (2015) consider that nurses could be driven to customers by a driver and serve some other customers on foot before being picked up. The resulting problem is similar to a dial-a-ride problem. Instances based on data from the Austrian red cross with up to 125 customers were solved with a Tabu Search heuristic.

The connection is also strong with the Two-Echelon VRP (Crainic et al., 2009), in which large vehicles transfer goods to smaller vehicles at so-called satellites, the later ones being allowed to deliver to the customer in a restricted delivery area. We refer the interested reader to Cuda et al. (2015) and Sluijk et al. (2022). Nonetheless, as in other routing problems with intermediate facilities (Guastaroba et al., 2016), the purpose of those is to concentrate the flows. Thus, only a few are considered especially because of their high initial cost in practice.

2.3. Discussion

We observe that, although problems with park-and-loop structures have already been investigated in the literature, none of these really consider the selection of parking spaces into a large set of available ones. The largest instances assume that each customer can be a parking space but we think that two considerations are worth investigating in a city logistics context: firstly, parking as close as possible to a customer may not be the best strategy depending on its accessibility and the direction from which the vehicle is coming. Secondly, a large set of areas are available for delivery vehicles in some cities and there is a need for efficient selection rules to be used in vehicle routing heuristics.

3. The Park-and-Loop Routing Problem with Parking Selection

This section introduces the Park-and-Loop Routing Problem with Parking Selection (PLRP-PS). Section 3.1 depicts the problem settings and Section 3.2 provides a mathematical model. For the sake of simplicity, we do not consider the time necessary to pick up packages at the vehicle nor the service duration at customers, but their integration in our model and method is straightforward.

3.1. Problem description

Let us consider a set N of customers with non-unitary demand q_i, $\forall i \in N$. Let $K = \{1, ..., k_{max}\}$ denote the homogeneous vehicle fleet, where k_{max} is the maximum vehicle fleet size. We consider that each vehicle is assigned to a driver who will perform deliveries by walking from a parking location to the customers. Accordingly, we indifferently refer to a vehicle $k \in K$ or to a driver $k \in K$ in this paper. All vehicles are based at a depot denoted o. The set P models the set of delivery areas. Each parking location $j \in P$ is associated with a parking time pt_j.

The PLRP-PS is defined on a directed graph $G = (V, A)$. The set of vertices is $V = N \cup P \cup \{o\}$. The set of arcs is $A = A_d \cup A_w$, where A_d is the set of driving arcs, with $A_d = \{(o,p) \mid \forall p \in P\} \cup \{(i,j) \mid \forall i,j \in P,\ i \neq j\} \cup \{(p,o) \mid \forall p \in P\}$, and A_w is the set of walking arcs, with $A_w = \{(p,i) \mid \forall p \in P,\ i \in N\} \cup \{(i,j) \mid \forall i,j \in N,\ i \neq j\} \cup \{(i,p) \mid \forall p \in P,\ i \in N\}$. In addition, we define the set $A_c = \{(i,j) \mid \forall i,j \in N, i \neq j\}$ of walking arcs between customers, with $A_c \subseteq A_w$. The driving time matrix is t^d , with t^d_{ij} $\forall (i,j) \in A_d$ the driving time for all driving arcs. A driving time t^d_{ij} integrates the parking time at destination pt_j.

The walking time matrix is t^w, with t^w_{ij} $\forall (i,j) \in A_w$ the walking time for all walking arcs. The walking distance matrix is d^w, with d^w_{ij} $\forall (i,j) \in A_w$ the walking distance for all walking arcs.

A *vehicle route* starts from the depot, visits a sequence of parking locations in P, and returns to the depot. The sum of customer orders served by a vehicle route must not exceed the vehicle capacity Q_d and a vehicle route duration must not exceed the value h_{max}. A *walking-trip* starts from a parking location in P, visits a sequence of customers, and returns to the same parking location. The duration of a walking-trip is the sum of the duration of all walking arcs used in this walking-trip. The customer orders served by a walking-trip should not be larger than the walking capacity denoted Q_w. The overall walking distance of each driver (i.e. the sum of the distance d^w_{ij} of the walking arcs used by a driver over all of its working trips) should not exceed the value w_{max}. At a parking location, a walking-trip can start after the vehicle is parked. Then, the drivers may perform several walking-trips from their parked vehicle.

The PLRP-PS consists of designing vehicle routes and walking-trips, with respect to their delivery and walking capacities, the maximum route duration of each vehicle, and the maximum walking distance of each driver such that each customer is served by exactly one driver. The objective function lexicographically minimizes the number of vehicles (z_1) and the sum of driving times, walking times, and parking times denoted z_2.

Figure 1 illustrates a PLRP-PS solution with $N = \{c_1, c_2, c_3, c_4, c_5\}$, $P = \{p_1, p_2, p_3\}$ and $K = \{k_1, k_2\}$. Vehicle routes are represented by solid lines and walking trips are represented by dashed lines.

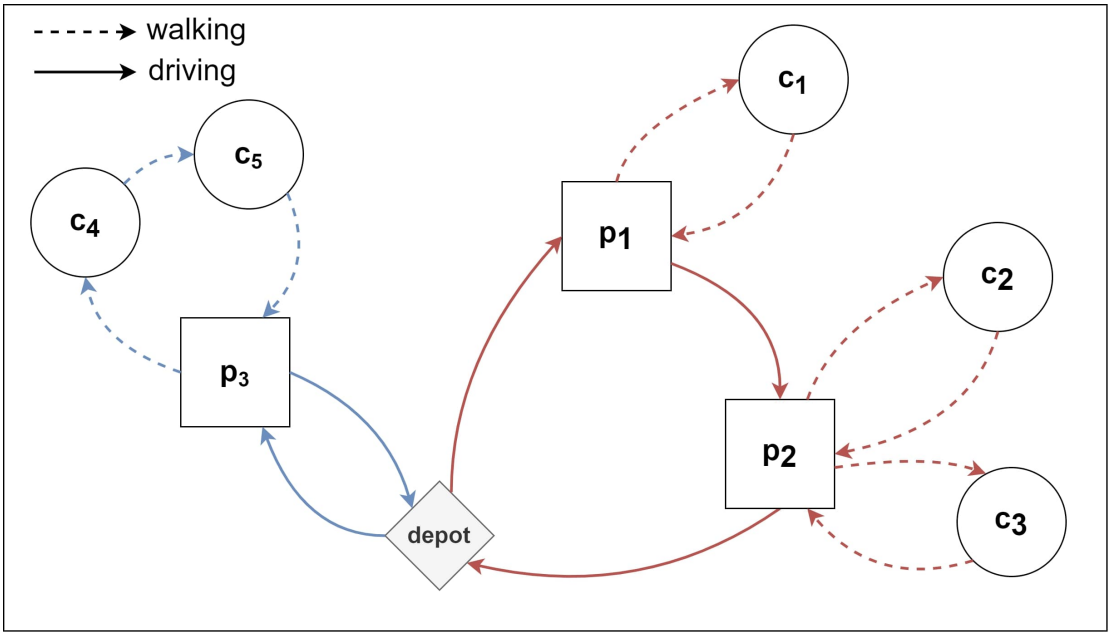


Figure 1: Routes and walking-trips representation in the PLRP-PS

3.2. Mathematical model

This section introduces an MILP formulation for the PLRP-PS. It introduces two classes of binary variables and two classes of real variables. First of all, x_{ij}^k is a binary variable that takes value 1 if arc $(i,j) \in A_d$ is used by vehicle $k \in K$, and is equal to 0 otherwise. Then, y_{ij}^{kp} is equal to 1 if arc $(i,j) \in A_w$ is used by driver $k \in K$ by walking while his/her vehicle is parked at $p \in P$, and 0 otherwise. The non-negative real variable r_p^k models the order in which a vehicle $k \in K$ (or its driver) visits parking locations $p \in P$. Finally, non-negative real variable u_i^k represents the remaining capacity of the driver $k \in K$ after visiting customer $c \in N$ during the considered walking-trip. According to this notation, the PLRP-PS can be modeled with the following MILP formulation:

$$lex - min(z_1, z_2) \tag{1a}$$

$$s.t.$$

$$z_1 = \sum_{k \in K} \sum_{i \in P} x_{0i}^k \tag{1b}$$

$$z_2 = \sum_{k \in K} \sum_{p \in P} \sum_{(i,j) \in A_w} t_{ij}^w y_{ij}^{kp} + \sum_{k \in K} \sum_{(i,j) \in A_d} t_{ij}^d x_{ij}^k \tag{1c}$$

$$\sum_{k \in K} \sum_{p \in P} \sum_{(i,j) \in A_w} y_{ij}^{kp} = 1 \qquad \forall j \in N \tag{1d}$$

$$\sum_{(i,j) \in A_w} y_{ij}^{kp} = \sum_{(j,i) \in A_w} y_{ji}^{kp} \qquad \forall k \in K, p \in P, j \in N \cup P \tag{1e}$$

$$\sum_{(i,p) \in A_d} x_{ip}^k = \sum_{(p,i) \in A_d} x_{pi}^k \qquad \forall k \in K, p \in P \tag{1f}$$

$$\sum_{(0,i) \in A_d} x_{0i}^k = \sum_{(i,n+1) \in A_d} x_{i,n+1}^k = 1 \qquad \forall k \in K \tag{1g}$$

$$\sum_{(i,p)\in A_d} x_{ip}^k \geq y_{pj}^{kp} \qquad \forall k \in K, p \in P, j \in N \qquad (1h)$$

$$y_{ij}^{kp} = 0 \qquad \forall k \in K, i, p \in P, i \neq p, j \in N \qquad (1i)$$

$$y_{ji}^{kp} = 0 \qquad \forall k \in K, i, p \in P, i \neq p, j \in N \cup P \qquad (1j)$$

$$\sum_{p\in P} \sum_{(i,j)\in A_w} d_{ij}^w \times y_{ij}^{kp} \leq w_{max} \qquad \forall k \in K \qquad (1k)$$

$$\sum_{p\in P} \sum_{(i,j)\in A_w} t_{ij}^w y_{ij}^{kp} + \sum_{(i,j)\in A_d} t_{ij}^d x_{ij}^k \leq h_{max} \qquad \forall k \in K \qquad (1l)$$

$$u_i^k \leq Q_w - q_i \times y_{pi}^{kp} \qquad \forall k \in K, p \in P, i \in N \qquad (1m)$$

$$u_j^k \leq u_i^k - q_j + (1 - y_{ij}^{kp})Q_w \qquad \forall k \in K, p \in P, (i,j) \in A_w \qquad (1n)$$

$$\sum_{p\in P} \sum_{(i,j)\in A_w} q_j \times y_{ij}^{kp} \leq Q_d \qquad \forall k \in K, j \in N \qquad (1o)$$

$$r_j^k - r_i^k \geq 1 - |P|(1 - x_{ij}^k) \qquad \forall k \in K, i, j \in P, i \neq j \qquad (1p)$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in K, (i,j) \in A_d \qquad (1q)$$

$$y_{ij}^{kp} \in \{0,1\} \qquad \forall k \in K, p \in P, (i,j) \in A_w \qquad (1r)$$

$$0 \leq u_i^k \qquad \forall k \in K, i \in N \qquad (1s)$$

$$0 \leq r_p^k \qquad \forall k \in K, p \in P \qquad (1t)$$

In this model, the objective function (1a) lexicographically considers two objectives. The first objective (1b) minimizes the vehicle fleet size, while the second objective (1c) minimizes the sum of walking time and driving time (that also includes parking time).

The first set of constraints (1d) fulfills customers' order satisfaction. Every customer must be served by a driver $k \in K$ using a walking arc $(i,j) \in A_w$ while the vehicle is parked at parking node $p \in P$. Constraints (1e) enforce flow conservation on walking arcs. A driver walking to a node $j \in N \cup P$ (either a parking or customer node) should also leave this node by walking. Constraints (1f) enforce the vehicles flow conservation. Constraints (1g) state that each vehicle in K must leave the depot and return to the depot. If a solution uses less than $|K|$ vehicles, then each unused vehicle $k \in K$ takes the arc $(0, n+1)$ with cost 0.

Constraints (1h) state that the driver of vehicle $k \in K$ can deliver $j \in N$ from parking node $p \in P$ if there is a driving arc x_{ip}^k that enters parking p. Constraints (1i) state that a driver cannot deliver a customer $j \in N$ directly from a parking node $i \in P$ if $i \neq p$ where $p \in P$ is the parking node assigned to the walking arc y_{ij}^{kp}. Constraints (1j) state that a driver cannot walk to a parking $i \in P$ if the car is parked at another parking $p \in P : i \neq p$.

The overall walking distance constraints (1k) are: for each vehicle $k \in K$ the sum of walking arcs distances used by this vehicle should not exceed w_{max}. Constraints (1l) limit the duration of a route, and its walking trips, to h_{max}, i.e. the day duration.

The walking capacity constraints are (1m) and (1n), the capacity remaining after delivering to the first client of a walking-trip $i \in N$ is given by (1m). Also, (1n) gives the capacity remaining after delivering the following clients in this walking-trip. The driving capacity constraints are (1o): for each vehicle $k \in K$ the sum of the demands served by this vehicle should not exceed the vehicle capacity Q_d. Finally, constraints (1p) rank the visits to parking. This adoption of the Miller-Tucker-Zemlin constraints (Dantzig et al., 1954) is used to eliminate subtours.

4. Small and Large Neighborhood Search

The *Large Neighborhood Search* (LNS) metaheuristic consists of iteratively destroying and repairing the current solution in order to improve it, until a time or iteration limit is reached. The destroy part appeals to so-called destroy operators and the repairing part to repair operators. The idea of LNS was first introduced by Shaw (1998) in constraint programming and reintroduced by Schrimpf et al. (2000) under the name "ruin and recreate." Its popularization as a metaheuristic is due to Ropke and Pisinger (2006a) who proposed an adaptive LNS in which several operators are selected depending on their previous performance. The method was shown to be efficient for solving large classes of VRP (Pisinger and Ropke, 2007; Ropke and Pisinger, 2006b). Recently, Christiaens and Vanden Berghe (2020) considerably improved the efficiency of the approach by introducing, among other features, a smaller destruction size and fast repair operators. Dumez et al. (2021a) combines small and large destruction sizes for the LNS. This approach is improved in Dumez et al. (2021b) to solve variants of the generalized VRP. The LNS variant of Dumez et al. (2021a) and Dumez et al. (2021b) was further detailed in Dumez (2021). In this paper, we present a more formal definition of this algorithm under the name *Small and Large Neighborhood Search* (SLNS).

The SLNS general algorithm is presented in Section 4.1. The destroy and repair operators designed to solve the PLRP-PS are described in Section 4.2, while the different types of insertions implemented in these operators are detailed in Section 4.3. Finally, the proposed parking selection strategies are presented in Section 4.4.

4.1. Algorithm

Let us first introduce the concept of partial solution and request bank, which is the LNS way to visit infeasible solutions. The LNS algorithm iteratively destroys and repairs solutions. Hence, the algorithm manipulates *partial solutions*. Given a solution S, the set $B(S)$, called the *request bank*, denotes the set of customers that are not served in S. When $B(S)$ is non-empty, S is a called *partial solution*. Partial solutions are used in operators, with the number of requests in the request bank penalized in the cost function as proposed by Pisinger and Ropke (2007). Accordingly, considering $z_2 = f(S)$, a modified objective function $z_2' = f'(S)$ is optimized in LNS, with:

$$f'(S) = f(S)\left(1 + \beta\frac{|B(S)|}{N}\right). \tag{2}$$

In SLNS, the small and large destruction sizes are randomly chosen in intervals $[\delta_{small}, \Delta_{small}]$ and $[\delta_{large}, \Delta_{large}]$, respectively. In addition, we presume two sets of destroy operators: denoted Σ^-_{small} and Σ^-_{large}, and two sets of repair operators denoted Σ^+_{small} and Σ^+_{large}. A *Small Neighborhood Search* (SNS) iteration draws a destruction size in $[\delta_{small}, \Delta_{small}]$, a destroy operator in Σ^-_{small} and a repair operator in Σ^+_{small}. A *Large neighborhood Search* (LNS) iteration draws a destruction size in $[\delta_{large}, \Delta_{large}]$, a destroy operator in Σ^-_{large} and a repair operator in Σ^+_{large}. Dumez et al. (2021b) propose to control these destruction sizes based on a parameter ω called the LNS frequency: if no new best solution is found within ω SNS iterations, then the next iteration will be an LNS iteration.

SLNS is described in Algorithm 1.

Algorithm 1 Small and Large Neighborhood Search

Input: δ_{small}, Δ_{small}, δ_{large}, Δ_{large}, Σ^-_{small}, Σ^-_{large}, Σ^+_{small},Σ^+_{large}, ω, β, S
Output: Best solution found S^*

1: $iter = 0$
2: **while** the time budget is not reached **do**
3: **if** $iter < \omega$ **then**
4: $iter \leftarrow iter + 1$ {small neighborhood search iteration}
5: $S' \leftarrow S$
6: randomly select a small destruction size $\Phi \in [\delta_{\text{small}}, \Delta_{\text{small}}]$
7: randomly select operators $\sigma^- \in \Sigma^-_{small}$ and $\sigma^+ \in \Sigma^+_{small}$
8: $S' \leftarrow \sigma^+(\sigma^-(S', \Phi))$
9: **if** $f'(S') < (1+\epsilon)f'(S^*)$ **or** $f'(S') < f'(S)$ **then**
10: $S \leftarrow S'$
11: **end if**
12: **else**
13: $iter = 0$ {large neighborhood search iteration}
14: $S' \leftarrow S^*$
15: randomly select a large destruction size $\Phi \in [\delta_{\text{large}}, \Delta_{\text{large}}]$
16: randomly select operators $\sigma^- \in \Sigma^-_{large}$ and $\sigma^+ \in \Sigma^+_{large}$
17: $S' \leftarrow \sigma^+(\sigma^-(S', \Phi))$
18: $S \leftarrow S'$
19: **end if**
20: **if** $f(S') < f(S^*)$ **and** $B(S') = \emptyset$ **then**
21: $S^* \leftarrow S'$
22: $iter \leftarrow 0$
23: **end if**
24: **end while**
25: **return** S^*

The iteration counter *iter*, initialized to zero in line 1, tracks the number of SNS iterations without improving the best known solution S^*. The counter *iter* is re-initialized to 0 on each LNS iteration (line 13) and each time S^* is improved (line 22). The main loop between lines 2 and 24 performs SNS iterations and LNS iterations as long as the time limit is not reached. On an SNS iteration ($iter < \omega$, Line 3 – 11), a new solution S' is created from the current solution S, a small destruction size ϕ is drawn in the interval $[\delta_{small}, \Delta_{small}]$, a small destroy operator σ^- is selected in Σ^-_{small} and a small recreate operator σ^+ is selected in Σ^+_{small}. The new solution S' is obtained (Line 8) by applying these two operators. S' is accepted as the current solution for the next iteration based on a record-to-record acceptance criterion (Dueck, 1993) applied on the modified cost function f' (line 9). In addition, a new solution is also accepted if it improves the current solution. This condition is used in particular on the SNS iterations that follow a LNS one.

If S^* is not improved for ω iterations, an LNS iteration is performed (line 13–19). In this case, the new solution S' is created from the best known solution S^*. The destruction size ϕ, destroy operator σ^- and repair operator are drawn from $[\delta_{large}, \Delta_{large}]$, Σ^-_{large} and Σ^+_{large}, respectively. The produced solution is then systematically accepted as the current solution for the next SLNS iteration. Over all iterations of the main loop, the best found solution S^* is saved (lines 20–22) and returned at the end of the algorithm.

As in Ropke and Pisinger (2006a) SLNS is used in two consecutive phases. The initial solution is found with an unlimited number of vehicles. The first phase reduces the number

of vehicles (z_1): every time a feasible solution is found (i.e. the request bank is empty), the maximum number of vehicles k_{max} is decremented, the route with the minimum number of customers is removed from the solution and its customers are added to the request bank. When the minimum number of vehicles $\left\lceil \frac{\sum_{i \in N} q_i}{Q_d} \right\rceil$ or the first phase time budget is reached, the second phase (minimization of z_2) starts.

4.2. Destroy and repair operators

This section introduces the SLNS operators that were implemented to solve the PLRP-PS. We will first briefly describe the **destroy operators** of the set Σ^-_{small} used in SNS iterations:

Random removal (Ropke and Pisinger, 2006a) randomly selects ϕ customers and removes them from the current solution.

Worst removal (Ropke and Pisinger, 2006a) removes ϕ customers one after the other, each time selecting the customer whose removal decreases the cost function the most.

Related removal (Hemmelmayr et al., 2012) selects a seed-customer randomly together with its $\phi - 1$ nearest customers. These are removed from the solution.

(Split) String removal (Christiaens and Vanden Berghe, 2020) removes sequences of customers in the walking trips of the current solution, either conserving, or not, a sub-string in the middle.

The destroy operators Σ^-_{large} used in LNS iterations bring diversification to the algorithm. Every small destroy operator described previously is also used as a large destroy operator, i.e. $\Sigma^-_{small} \subset \Sigma^-_{large}$.

In addition to the original versions, we adapt these four operators to the PLRP-PS. To take this problem structure into account, some destroy operators can be either applied to customers or to parking locations (similarly to Hemmelmayr et al. (2012)). Hence, for each operator we create an equivalent destroy operator that removes parking locations instead of customers.

When a visited parking location is removed from a solution, all the customers that are served by the walking trips starting from this parking spot are placed in the request bank. For example, the *random parking removal* operator randomly selects some visited parking locations, all of their visited customers are placed in the request bank and the visit to that particular parking spot is removed from the route, until ϕ customers have been removed. Symmetrically, when a walking trip becomes empty (after removing its last customer), the corresponding parking location is removed from the vehicle route.

Historical removal (Pisinger and Ropke, 2007) is adapted here to focus on removing parking locations exclusively. It is based on the best objective value obtained by a previous solution that was using a particular parking location. For each parking location, the best objective function value (z_2) is stored over SLNS iterations. This operator iteratively removes the parking location with the worst objective value in the history until at least ϕ customers have been placed in the request bank.

Route removal (Hemmelmayr et al., 2012) removes a random route from the solution.

In our SLNS implementation, the same set of **repair operators** Σ^+ is used for large and small destruction sizes. Let S be a partial solution and $B(S)$ the set of customers in the request

bank. A repair operator $\sigma^+ \in \Sigma^+$ evaluates the insertion of customers in $B(S)$. These are selected according to list heuristics, as proposed in Christiaens and Vanden Berghe (2020):

Random selects customers in a random order.

Closest selects customers whose nearest parking location is the closest to the depot first.

Farthest selects customers whose nearest parking location is the farthest to the depot first.

Largest selects customers with the largest demand first.

First in first out (FIFO) selects customers according to their order of insertion in $B(S)$.

When a customer is selected for insertion, it has to be inserted in a walking-trip, starting and returning from a parking location on a route. The insertion position process with parking selection is detailed in Section 4.3). It may happen that no feasible insertion position is found for a customer. In this case, it remains in the request bank and the next customer is selected. This results in a partial solution.

Note that in all repair operators, a parking spot already used by a vehicle is never inserted into another route or in another position on its current route.

Table 1 presents an overview of the operators that are implemented in the SLNS algorithm. Sets in columns *Customer/Parking/Route* indicate when they may be used depending on what they operate on. The references in the column *source* indicate the articles that proposed the considered criteria.

Type	Name	Customer	Parking	Route	Source
Destroy	Random	$\Sigma^-_{small}, \Sigma^-_{large}$	Σ^-_{large}		Ropke and Pisinger (2006a)
	Worst	$\Sigma^-_{small}, \Sigma^-_{large}$	Σ^-_{large}		Ropke and Pisinger (2006a)
	Related	$\Sigma^-_{small}, \Sigma^-_{large}$	Σ^-_{large}		Hemmelmayr et al. (2012)
	String	$\Sigma^-_{small}, \Sigma^-_{large}$	Σ^-_{large}		Christiaens and Vanden Berghe (2020)
	Historical		Σ^-_{large}		Pisinger and Ropke (2007)
	Route			Σ^-_{large}	Hemmelmayr et al. (2012)
Repair	Random	$\Sigma^+_{small}, \Sigma^+_{large}$			Christiaens and Vanden Berghe (2020)
	Closest	$\Sigma^+_{small}, \Sigma^+_{large}$			Christiaens and Vanden Berghe (2020)
	Farthest	$\Sigma^+_{small}, \Sigma^+_{large}$			Christiaens and Vanden Berghe (2020)
	Largest	$\Sigma^+_{small}, \Sigma^+_{large}$			Christiaens and Vanden Berghe (2020)
	FIFO	$\Sigma^+_{small}, \Sigma^+_{large}$			*this paper*

Table 1: Overview of destroy and repair operators

4.3. Customer insertion

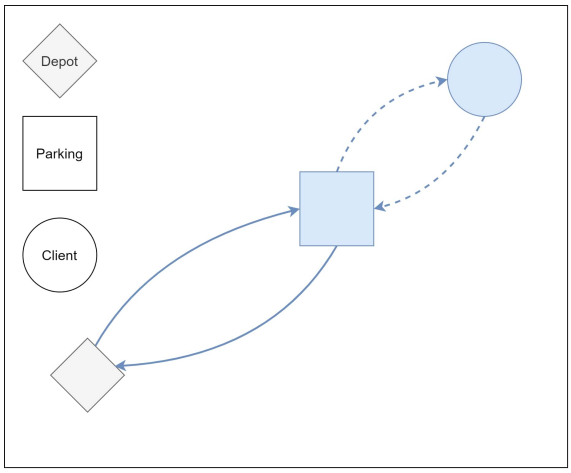
When a customer has been selected to be inserted in a solution, three types of insertions are considered:

- **Insertion in a new route:** the first customer is inserted into a solution by creating a *new route* (Figure 2a). Hence, in this insertion a customer is served from a new route and a new parking location.

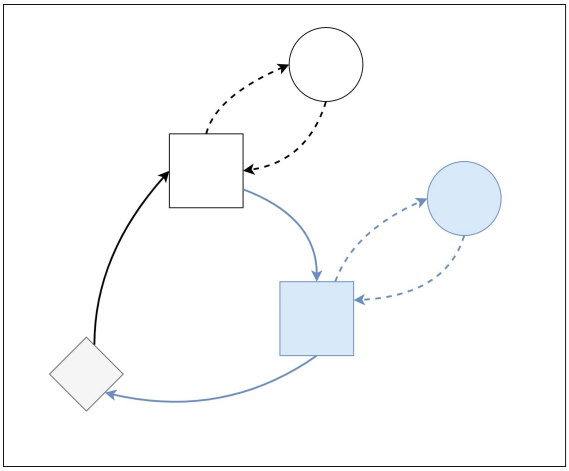
- **Insertion through a new parking location:** a customer can be inserted into a new walking-trip, starting from a *new parking location* (Figure 2b) on an existing route. This operation adds a parking location to the list of used parking spots.

- **Insertion through a served parking location:** both of the above types of insertions involve stopping at a new parking location, while serving a customer from a *served parking location* takes advantage of the vehicle being already parked to serve several customers. In this case, all possible insertions on existing walking-trips (Figure 2c), or on new walking-trips from each served parking spot (Figure 2d) are evaluated.

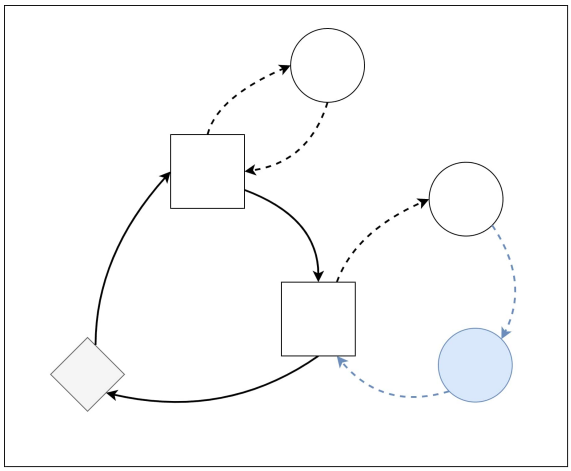
For each customer insertion, for all insertion types and unless a parking selection strategy applies (as presented in Section 4.4), all insertion positions (all parking and customer positions in walking trips) are evaluated and the best one is inserted. Note that in this process a diversification is obtained by integrating the *blink* principle of Christiaens and Vanden Berghe (2020): insertions are evaluated with a probability $1 - \gamma$, i.e. they have a probability γ to be ignored. The performed insertion is the cheapest one among all evaluated insertions.



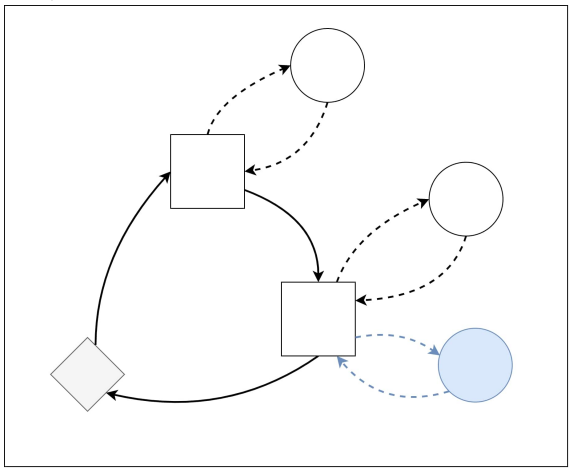
(a) Insertion in a new route



(b) Insertion through a new parking location (existing route)



(c) Insertion in an existing walking-trip (existing route, served parking location)



(d) Insertion in a new walking trip (existing route, served parking location)

Figure 2: Types of insertions in PLRP-PS solutions

4.4. Parking location selection strategies

Inserting a customer in a new route or through a new parking location (Figure 2a and 2b) suggests selecting a parking location among a very large set of potential locations. The proposed selection strategies are applied to reduce the number of evaluated parking spots in these types of insertions.

Let us presume that a customer has been selected and our goal is to evaluate its insertion in a new route or through a new parking location. We propose different parking selection strategies. A *parking selection strategy* is composed of a *parking selection criterion* and *parking selection method*.

We consider two criteria to select parking locations from each customer: The first criterion is the walking time between the parking location and the customer (*radius-r* criterion). The second criterion is the ranking of the parking spot in the list of parking spots ordered by non-decreasing distance to the considered customer (*k-nearest* criterion). In addition, two methods involve these criteria to reduce the set of evaluated parking spots: The first one filters the parking set from which each customer can be served a priori, keeping the locations below a given threshold on the chosen criterion. In the second method, for each customer a sorted list of parking locations is designed based on the selected criterion. The locations are then evaluated according to this order, with a given probability to stop after each location evaluation. These two methods are detailed below.

4.4.1. Filtering parking locations

Let us presume that, for each customer $i \in N$, a subset of parking locations is selected from P and is referred to as $P_i \subset P$. According to the *filtering* method, when the customer i has to be inserted in a new route or with a new parking location, the location j is considered if and only if $j \in P_i$.

Given a parameter $r > 0$, the radius-r criterion defines P_i as the subset of parking spots within r minutes walking time from i, i.e. $P_i = \{j \in P | t^w_{ij} \leqslant r\}$. The k-nearest criterion, with $k \leq 1$ defines P_i as the subset containing the k nearest parking locations, walking from i.

Note that in a solution, a customer i may still be served from a parking location not included in P_i if it has been inserted in an existing trip or through a served parking location.

4.4.2. Sorting parking locations

The second method is denoted *sorting*. It consists of building, for each customer, a sorted list of parking locations. With both criteria, for each customer $i \in N$, the parkings $j \in P$ are sorted in non-decreasing order of t^w_{ij}. When inserting a customer i in a new route or through a new parking location, the parking locations are considered according to their order in the customer's sorted list of parking spots.

For the radius-r criterion, for a given parameter r, the list of parking locations is explored while $\mu \leq \exp\left(-t^w_{ij}/r\right)$ where $\mu \sim U([0;1])$ and t^w_{ij} is the walking time between the customer i and the parking spot j. As for the simulated annealing criterion (Kirkpatrick et al., 1983), this criterion is designed such that the expected walking time between i and the last parking location explored is around r minutes when the exploration stops.

For the k-nearest criterion, for a given parameter k, the list of parking locations is explored while $\mu \leq 1 - \frac{1}{k}$, with $\mu \sim U([0;1])$. Let Y be a random variable modeling the number of evaluated parking locations. By definition of a geometric distribution, we can say that $Y \sim Geo(\frac{1}{k})$, and that its expected value is $E(Y) = k$. In other words, the expected position of the parking location in the list when the exploration stops is k.

5. Computational Results

This section presents the experiments performed with SLNS to solve the PLRP-PS. The algorithm is coded in Julia 1.5.0. The experiments were performed on a PC running Linux, Ubuntu 20.04.2 LTS, equipped with an Intel Xeon Gold 6230 @ 2.10GHz. A single thread is used by our code.

Section 5.1 introduces the instances that were generated for these experiments based on realistic data. Section 5.2 compares the different parking selection strategies. SLNS is evaluated on the PLRP, and our results are compared to those of Coindreau et al. (2019) and Cabrera et al. (2022) in Section 5.3. Finally, numerical studies are performed in Section 5.4 to evaluate the impact of various delivery area configurations.

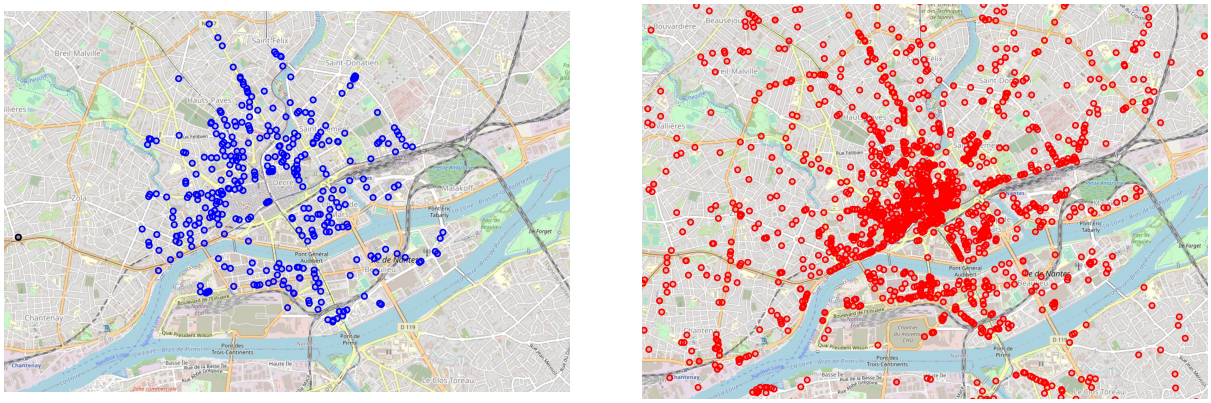
5.1. Instances and parameters

We created realistic instances based on open data from Nantes Metropole [1], as well as on the official French open data platform [2].

The set of parking locations is taken from the "Loading zones in downtown Nantes" dataset [3] (Figure 3a). The customer locations were obtained with filters: "active," "accommodation," and "catering locations" in "Loire-Atlantique" on the dataset "SIRENE database of companies in Nantes Metropole" [4] (Figure 3b).

Our full dataset encompasses 352 loading zones (Figure 4a) and 1196 customer locations within walking range of a loading zone (Figure 4b). The depot, represented by the black circle on the left of Figure 4b, is located outside of the city center.

Real-world distances and travel time matrices were obtained with openrouteservice backend [5]. The driving speeds are heuristically computed by this software based on the legislation and road characteristics. A walking speed of $4.8km/h$ is assumed. All visualizations were created using Open Street Map [6].



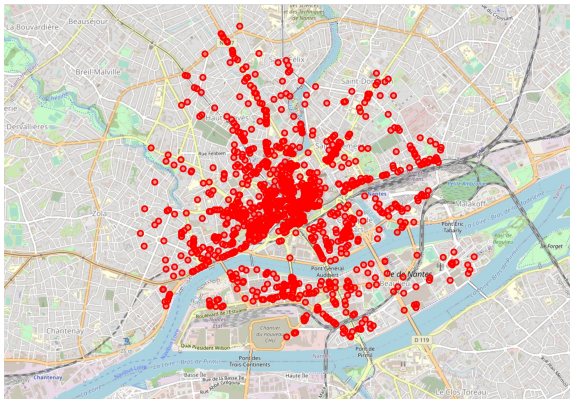
(a) Loading zones

(b) All customers locations

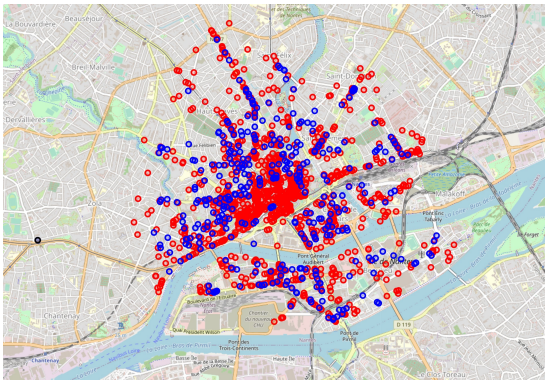
Figure 3: Datasets representation

This data set is used to generate five sets of instances, including between 50 and 400 customers. For each instance size, ten instances were generated, taking customer locations at

[1] `https://data.nantesmetropole.fr`
[2] `https://www.data.gouv.fr`
[3] `https://www.data.gouv.fr/fr/datasets/aires-de-livraison-du-centre-ville-de-la-ville-de-nantes/`
[4] `https://data.nantesmetropole.fr/explore/dataset/244400404_base-sirene-entreprises-nantes-metropole/`
[5] `www.openrouteservice.org` and `www.github.com/GIScience/openrouteservice`
[6] `www.openstreetmap.org`



(a) Reduced customers locations



(b) Loading zones + reduced customers locations

Figure 4: Reduction of the datasets

random in the set of 1196 locations, and customers demands are randomly generated integer values between 1 and 5. The complete set of loading zones is considered in all instances. The considered time horizon is 420 minutes (i.e 7h). In our experiments, we set the service duration at customers to 0.

Figure 5 illustrates a solution of the PLRP-PS, showing the real driving and walking paths that are considered in this dataset. The red points indicate customers and the dark blue points indicate the selected commercial loading zones. Lightblue points indicate loading zones that are not used. Walking paths are represented in red, while the other colors show the vehicle itineraries.

For the sake of readability, driving paths between the depot and parking locations or loading zones are replaced by dashed lines. Driving and walking paths are obtained with the openroute-service backend[5].

Table 2 summarizes the default parameter values taken for SLNS and for the PLRP-PS that were used in our experiments. The parameters of the parking selection strategies will be thoroughly investigated in Section 5.2. The time budget used in SLNS for each instance size $n = \{50, 100, 200, 300, 400\}$ is $\{1, 3, 10, 20, 40\}$ minutes, respectively.

PLRP-PS:	
k_{max}	∞ (to be minimized)
Q_d	50
Q_w	10
pt	5 minutes
w_{max}	8 kilometers
h_{max}	420 minutes (i.e 7h)

SLNS:	
initial solution	Largest repair operator
$[\delta_{small}, \Delta_{small}]$	$[0.05\|N\|, 0.10\|N\|]$
$[\delta_{large}, \Delta_{large}]$	$[0.10\|N\|, 0.20\|N\|]$
ω	$\|N\|^{1.5}$
β	20
γ	10%

Table 2: Default SLNS and PLRP-PS parameter values

5.2. Selection of parking locations

This section provides insights into the main focus of this paper: the selection of parking locations in SLNS. In this section, we present different results of SLNS on the PLRP-PS, focusing on the parking selection parameters introduced in Section 4.4, comparing the filtering and sorting methods as well as the k-nearest and radius-r criteria.

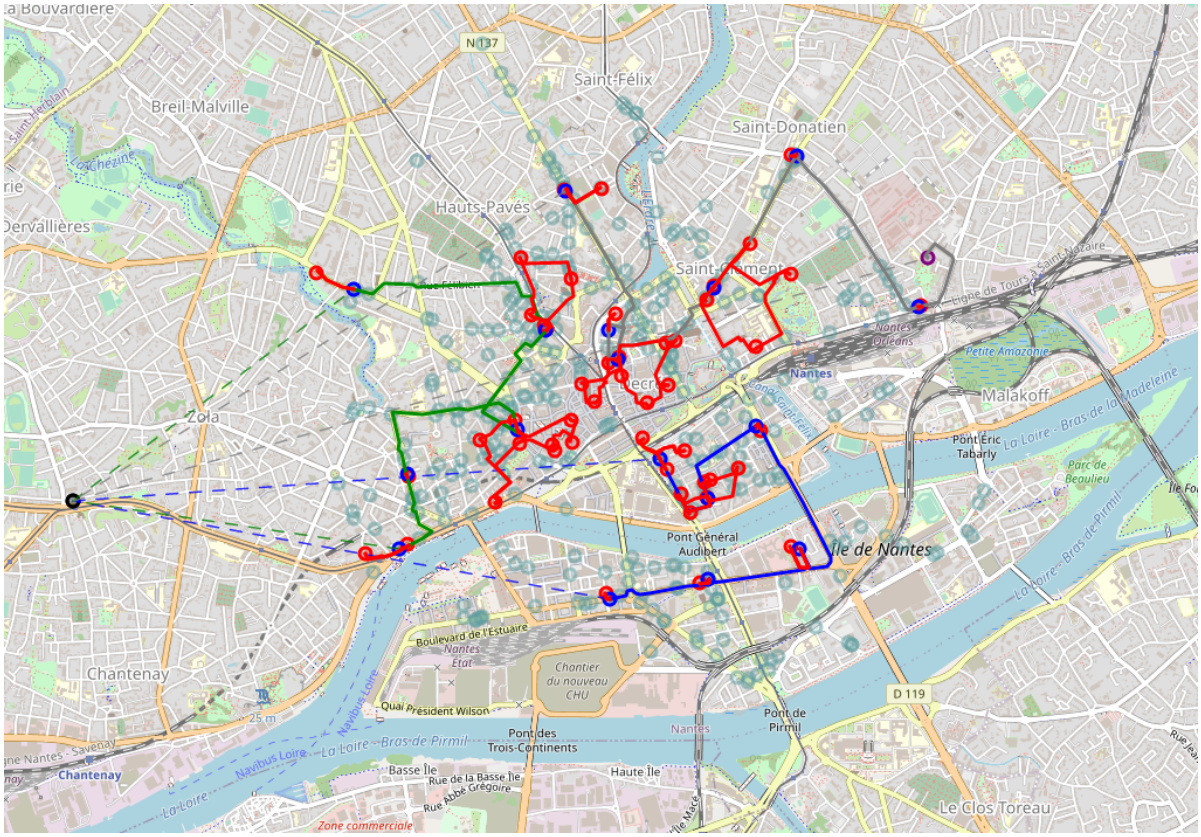


Figure 5: Visualization of the best found solution for instance "nantes-50-1" with 50 customers.

To find the best parking selection method, the different strategies were tested with different parameter values. In addition, as a base case, we consider the *no-filter*(x5) configuration in which the maximum runtime is multiplied by five and no selection method is applied (all parking locations $p \in P$ are evaluated in all types of insertion). Then, the results of these configurations are compared to the best-known solutions (BKS) with different measures: number of BKS found (i.e gap is zero), average gap, and maximum gap. The gap is computed such as $\Delta = \frac{z-BKS}{BKS} \times 100$, where z is the sum of driving time, walking time, and parking time.

Table 3 synthesizes the results of these experiments (the detailed results are provided in Appendix in Table A.7).

For each configuration, all instances are solved five times and the best solution out of these five runs is kept. Each line corresponds to a configuration. The configurations are first grouped by strategy: k-nearest or radius-r. Secondly, they are grouped by approach: filtering or sorting. Then, Column 3 specifies the value of r (in minutes) or k (in number of parking locations). Finally, Columns 4, 5, and 6 give the number of best known solutions found, the maximum gap, and the average gap, respectively.

From these experiments we find that the *k-nearest* parking selection strategy outperforms the *radius-r* strategy. An interpretation is that the duration radius r integrates too many parking locations for customers in dense areas, and not enough when customers are isolated. On the contrary, the k-nearest criterion considers a limited and controlled number of parking locations for customers regardless of the density of parking locations. To understand the poor performance of the filtering strategy with r=5 or r=10, let us have a deeper look into the solution of instance nantes-50-1. For the $r = 5$ radius criterion with the filtering parking selection method, the best solution has a total working time of 394.1, with a driving time of

Parking selection strategy			# BKS	Max. Δ	Avg. Δ
radius-r		r			
	filtering	5	0	38.67	19.22
		10	0	28.56	15.93
		15	1	5.45	1.90
	sorting	5	1	5.35	1.93
		10	3	5.40	2.11
		15	0	7.64	2.59
k-nearest		k			
	filtering	5	4	3.48	0.88
		10	**13**	2.90	0.88
		15	3	4.08	1.34
	sorting	3	8	3.37	0.81
		5	6	**2.37**	**0.79**
		10	7	2.85	0.85
	no-filter(x5)	-	5	3.12	1.25

Table 3: Comparison of the parking selection strategies

80.4, a walking time of 258.7 and a parking time of 55. On the other hand, for the $k = 10$ nearest criterion with the filtering method, the best solution has an overall working time of 326.5 with a driving time of 78.8, a walking time of 157.7, and a parking time of 90. With too few possibilities to start new walking trips from new parking locations, the insertion algorithm mainly inserts in existing trips, resulting in a solution with long walking trips.

As far as the *k-nearest* parking selection criterion is concerned, the *filtering* method with $k = 10$ and the *sorting* method with $k = 5$ both perform well. Let us denote them *filtering-k10* and *sorting-k5*, respectively. The sorting-k5 strategy has better average and maximum gaps, but filtering-k10 seems to provide more best-known solutions.

Figure 4 presents a more detailed comparison of these two strategies. The results are aggregated per instance size. In this table the column #Best indicates the number of instances for which each strategy allows one to find the best solution out of the solutions given by the two strategies. An additional column #veh indicates the average number of vehicles in each category of instance. The first thing to notice here is that, when compared only to filtering-k10, the sorting-k5 strategy is often better than filtering-k10. This comforts the slight superiority of sorting-k5 over filtering-k10 in our experiments. Nevertheless, the two strategies remain very close. In particular, they find the same average number of vehicles per category.

According to these results, the following experiments are performed using the sorting-k5 strategy.

5.3. Computational experiments on benchmark PLRP instances

The performance of the proposed SLNS is evaluated in the PLRP, in which the set of parking locations is restricted to the set of customers. PLRP instances can be obtained from the VRP-TR instances of Coindreau et al. (2019), without car-pooling and time windows. These

	filtering-k10				*sorting-k5*			
n	#Best	MaxΔ	AvgΔ	#veh	#Best	MaxΔ	AvgΔ	#veh
50	4	**0.71**	**0.17**	**3.7**	**6**	1.21	0.2	**3.7**
100	**5**	**1.07**	**0.31**	**6.7**	**5**	1.74	0.35	**6.7**
200	3	2.9	0.72	**12.3**	**7**	**2.02**	**0.4**	**12.3**
300	4	1.9	0.84	**18.4**	**6**	**1.64**	**0.5**	**18.4**
400	**6**	**2.2**	**0.48**	**24.5**	4	2.37	0.61	**24.5**
Avg.	22	2.9	0.5	–	**28**	**2.37**	**0.41**	–

Table 4: Detailed comparison per instance size of the filtering-k10 and sorting-k5 strategy.

instances are available online [7] and they have been solved under these assumptions by the VNS of Coindreau et al. (2019) and the matheuristic of Cabrera et al. (2022).

Coindreau et al. (2019) minimize the sum of driving distances. To adapt our method, the time needed to park is fixed to zero and the walking cost is obtained by multiplying all walking distances by a small value ε. The constraints used by Coindreau et al. (2019) to solve their instances are as follows: a maximum overall walking distance for each worker of 5 km (here named $w_{max} = 5$) and a time horizon of 7h (here $h_{max} = 420$). These instances also contain a service duration at each customer. This service duration is not considered as a part of the objective function but it is used to evaluate the day duration constraint.

For this set of experiments, the time limit of SLNS is set to 15, 30, 60, and 120 seconds for the 20, 30, 40, and 50 customers instances, respectively. These values are chosen very close to the average run-times of Cabrera et al. (2022), who used a limited number of iterations. The average run-times of Coindreau et al. (2019) are significantly greater (6779 seconds for 50 customer instances). Ten runs are performed for each instance.

Table 5 synthesizes the comparison between the proposed SLNS algorithm, the VNS of Coindreau et al. (2019), and those of the matheuristic of Cabrera et al. (2022) (denoted MH). The detailed results for each instance can be found in Table A.6. Similarly to the previous tables, each line represents averaged results for each instance size. For each method, the columns indicate the number of BKS (including our solutions), the maximum gap to the best known solution (in %), and the average gap to the best-known solutions (in %).

	VNS			MH			SLNS		
n	# BKS	Max. Δ	Avg. Δ	# BKS	Max. Δ	Avg. Δ	# BKS	Max. Δ	Avg. Δ
20.0	1.0	8.89	4.11	8.0	0.1	0.02	10.0	0.0	0.0
30.0	1.0	5.85	3.13	9.0	0.04	0.01	7.0	0.6	0.16
40.0	3.0	5.4	1.16	7.0	0.59	0.08	9.0	1.97	0.2
50.0	3.0	6.18	1.99	4.0	2.57	0.77	7.0	1.31	0.2
Total/Avg.	8.0	6.58	2.6	28.0	0.83	0.22	33.0	0.97	0.14

Table 5: Comparison on the Coindreau et al. (2019) PLRP instances to the VNS of Coindreau et al. (2019) and the matheuristic (MH) of Cabrera et al. (2022).

From these experiments we find that, with equivalent computing times, SLNS finds a little more BKS than MH though it is not always on the same instances. New BKS are found for eleven instances. Overall, SLNS misses only seven BKS over the set of forty instances. In

[7] `https://github.com/ncabrera10/VRPTR_instances`

addition, SLNS has the smallest average gap (0.14%) on average, while MH has a slightly better average for maximum gaps. It can be noted that with slightly longer run-times (600 seconds on all instances), the SLNS average of average gaps falls to 0.05, while the average of maximum gaps falls to 0.44.

In conclusion, SLNS offers competitive results on the PLRP. With equivalent run-times, the results slightly outperform the elaborate matheuristic of Cabrera et al. (2022). It also has the advantage of being simple to design and easy to integrate in classical VRP heuristics.

5.4. Managerial insights

The objective of this section is to evaluate the impact of integrating parking selection and walking trips in vehicle routing. Therefore, we present computational experiments in which the PLRP-PS is solved with different parameter values and compared with the CVRP and PLRP. These experiments are led on the Nantes instances presented in Section 5.1. We evaluate the impact on driving time, walking time, and parking time as well as the number of times vehicles park as the number of possible parking location increases.

As described in the previous section, our SLNS for the PLRP-PS can solve the PLRP by limiting the set of parking locations to the set of customers ($P = N$). It can also be used to solve a CVRP by taking $P = N$ and setting the maximum walking distance per driver, w_{max} to zero.

5.4.1. Impact of different compositions of the parking locations set

In the following experiments, we compare the solutions obtained with SLNS when solving the Nantes instances as CVRP, PLRP, or as PLRP-PS. In the latter case, we consider two sets of parking locations: either the set of loading zones (denoted LZ), or the set made of parking at each customer location, plus the loading zones (denoted C+LZ). The parking time pt is set to five minutes in all cases. Figure 6 presents a breakdown of the objective function value between driving time, parking time, and walking time, on average for each PLRP-PS instance size, divided by the number of customers, for the different problem settings.

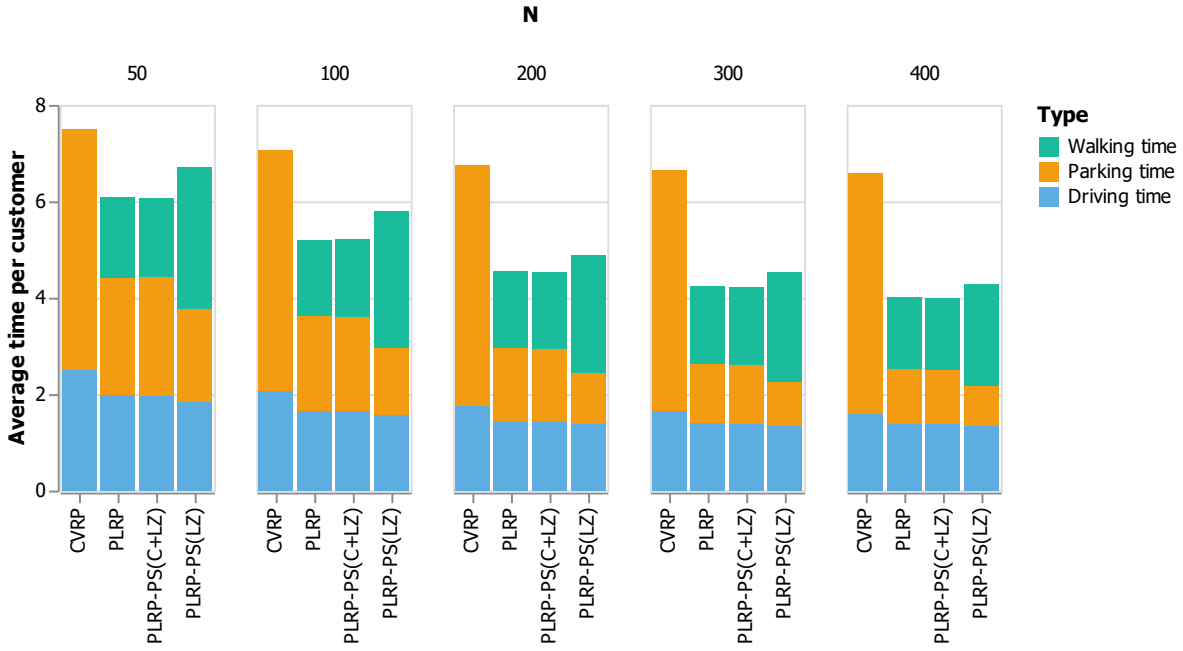


Figure 6: Decomposition into driving, parking, and walking time

The first observation is that all PLRP variants achieve a reduction in cost with respect to the CVRP. In more detail, the PLRP solutions reduce driving time by 16% and parking times by 72% compared to the CVRP. Moreover, the PLRP solution costs are on average 35% cheaper. This indicates that, even when parking at customers in city centers takes reasonable time, walking to serve several customers from one parking location should be considered as a way to reduce cost. Comparing the PLRP-PS(LZ) to the CVRP, the driving time is reduced by 19%, the parking time is reduced by 80% and the objective value is reduced by 30%. The difference of reduction of the objective value between the PLRP and PLRP-PS(LZ) is due to the walking time. In the latter case, the average walking time is 50% greater than for the PLRP. Still looking at the overall cost, it is interesting to note that the relative cost saving is greater for the PLRP variants when the number of customers is high. This indicates that park-and-loop efficiency increases with the density of customers, which was expected.

Secondly, the results for the PLRP are very similar to the ones with PLRP-PS(C+LZ) where all customers and delivery areas are considered parking locations. These results indicate that it is more efficient to park at customers when it is possible. Looking at the detail of driving, parking, and walking times, parking at customers mainly reduces the walking time while driving and parking times are higher.

To further analyze the PLRP and PLRP-PS solutions, Figure 7 represents the average number of parking instances per customer (i.e. the number of used parking locations divided by the number of customers) for each instance size and each PLRP variant.

Parking at customers is represented in orange, while parking at loading zones is in green. Let us recall that some parking at customers have been added to the PLRP-PS(LZ) parking locations for customers that are too far from a loading zone.

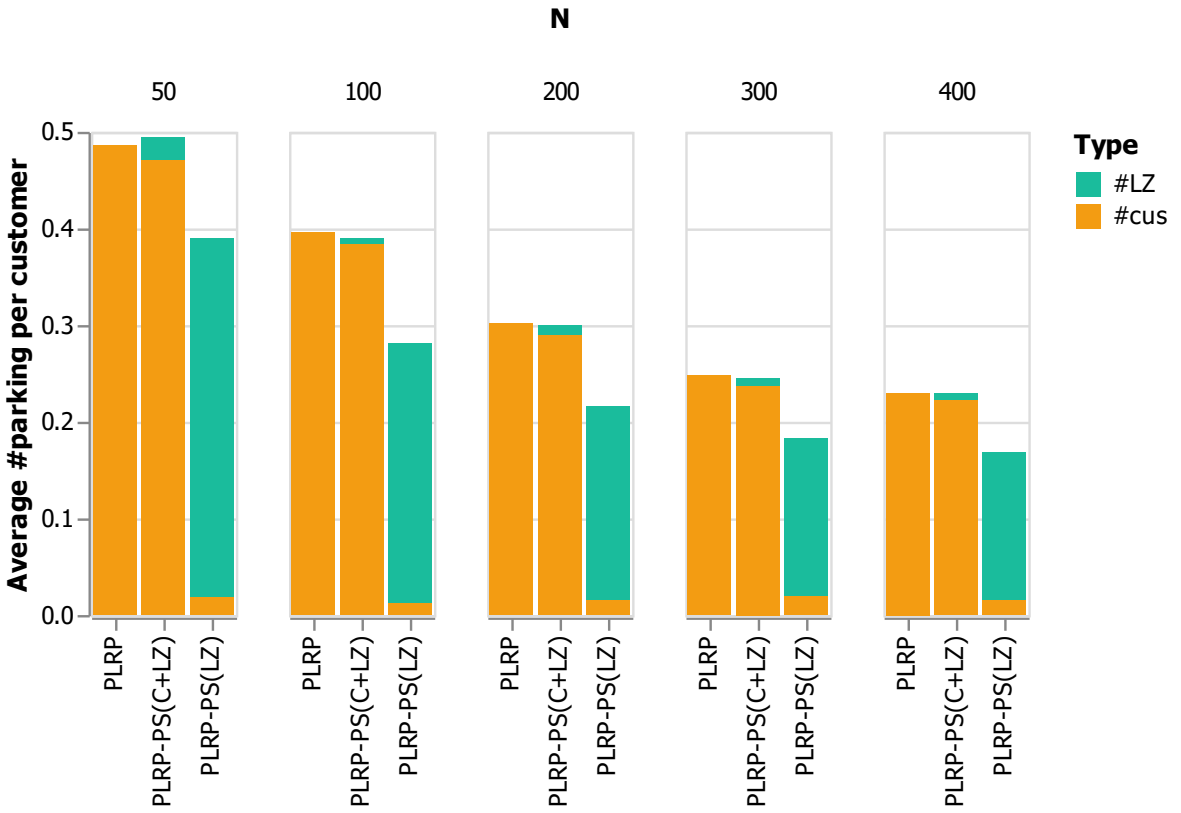


Figure 7: Average number of parking instances per customer and per parking type in the PLRP and PLRP-PS solutions

The figure confirms that parking at customers is mostly used in PLRP-PS(C+LZ) solutions. When parking farther from customers (i.e. in PLRP-PS(LZ) solutions), fewer parking locations are used.

It is also worth noting that the average number of parking instances per customer decreases when the number of customers increases. This confirms that the greater the density of customers, the more efficient the park-and-loop practice.

5.4.2. Impact of parking time

When comparing the different variants of the PLRP-PS, there is no doubt that parking time is of uttermost importance. To evaluate its impact, we compare the solutions of the PLRP-PS with different values of $pt \in \{0, 3, 5, 15\}$ (in minutes). The parking set is the set of loading zones (PLRP-PS(LZ)). Figure 8 shows the impact of parking time variations.

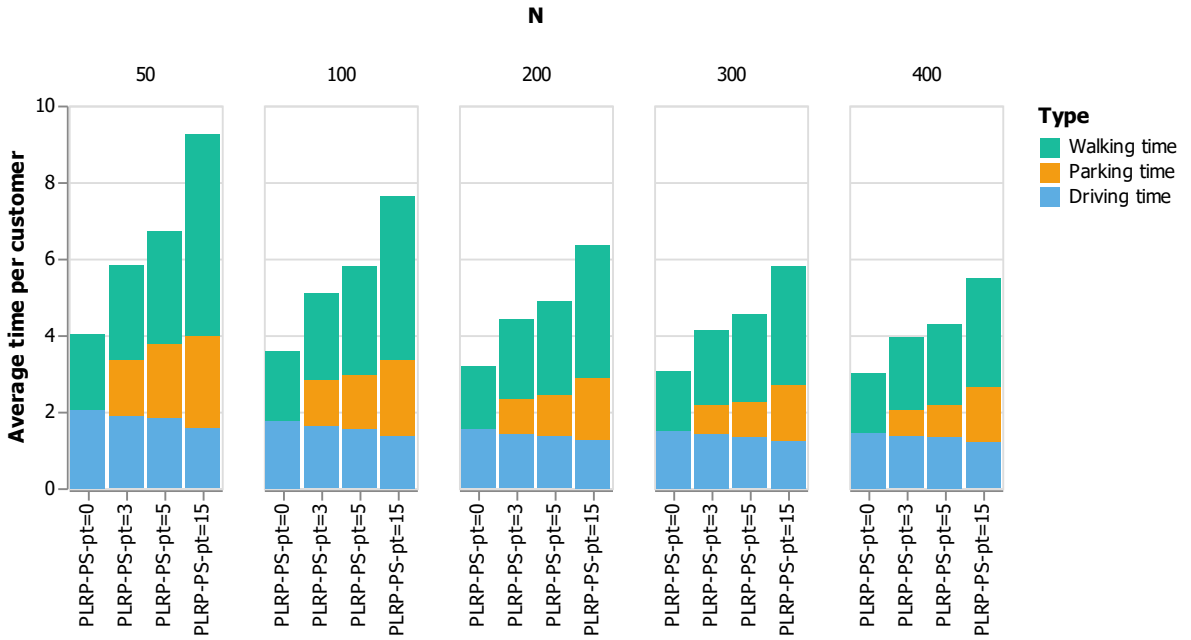


Figure 8: Average time per customer decomposed by driving time, parking time, and walking time

For each value of pt and each instance size, the bar chart shows the average driving time, walking time, and parking time per customer. A first observation is that for all instance sizes, the average driving time per customer remains relatively stable when pt increases. The average parking time per customers in solutions also increases with pt but this increase remains consequently smaller than the increase of pt itself. On the contrary, the walking time significantly increases, especially on small instances. On average, when parking time increases from $pt = 0$ to $pt = 5$, the average driving time per customer is reduced by 10% and the walking time is increased by 44%. Looking at the impact on the overall working time, increasing the parking time from $pt = 0$ to $pt = 5$ generates a limited increase of working time by less than three minutes per customer for the least favorable instances ($N = 50$). As far as the 400-customers instances are concerned, this increase is even reduced to one minute and seventeen seconds on average. This confirms that considerable savings can be generated by the park-and-loop practice when parking requires significant time.

Looking at the number of parking instances per customer on Figure 9, we observe that when pt increases, the proportion of parking locations used decreases. For instance, when $pt = 5$ minutes, the average number of parking locations used in solutions is reduced by 59%

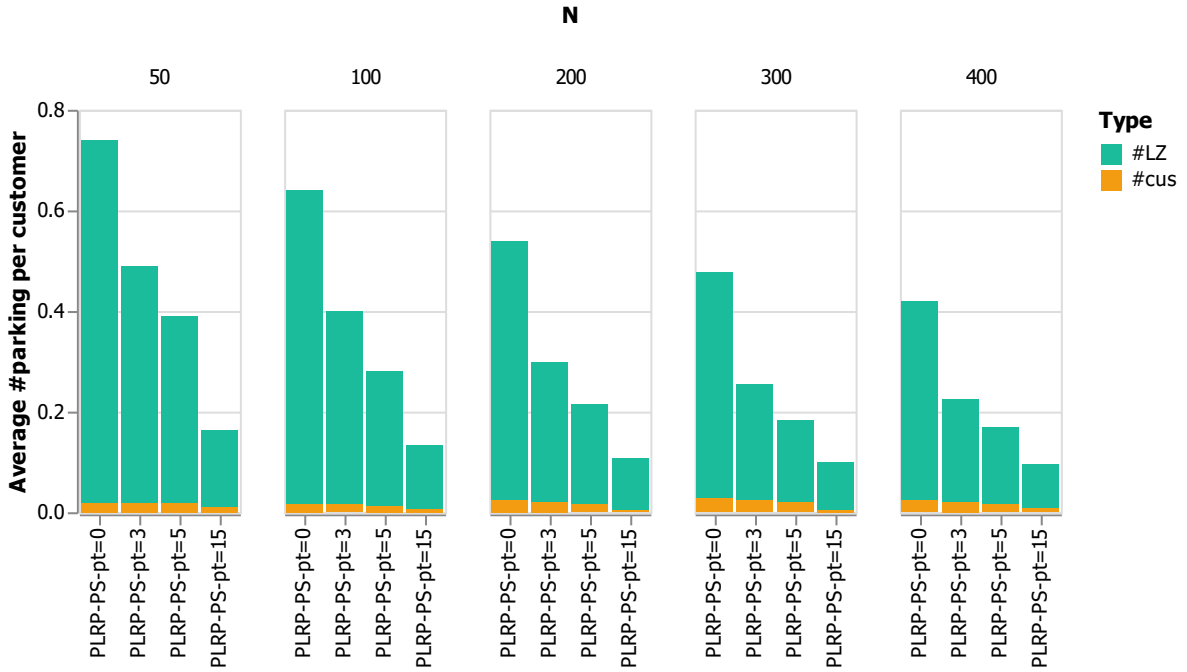


Figure 9: Evolution of the average number of parking instances per customer with respect to parking time

compared to solutions where $pt = 0$. It is worth observing that even when the parking time is null, a parking spot is generally used to serve several customers. This can be explained by the fact that not every customer is necessarily in the immediate vicinity of a loading zone.

6. Conclusion

In this paper, we introduced the Park-and-Loop Routing Problem with Parking Selection (PLRP-PS), which is an extension of previous vehicle routing problems where customers are served through a combination of walking and driving, presented under the denomination Park-and-Loop Routing Problem (PLRP). The PLRP-PS captures practical aspects of urban delivery by integrating the selection of a parking location into vehicle routing. To solve this problem, we proposed a Small and Large Neighborhood Search (SLNS) metaheuristic combining recent progress in the large neighborhood search meta-heuristic. We led extensive computational experiments that focus on the integration of parking selection strategies in vehicle routing heuristics. We compared four parking selection strategies which are integrated in the repair operators of SLNS. These strategies are evaluated in real-world instances built upon public data from the city of Nantes.

We find that selection strategies based on the selection of a number of nearest parking locations for each customer are more efficient than those based on the selection of parking spots within a fixed radius around each customer. Despite its simplicity, we show that SLNS competes with state of the art metaheuristic on existing benchmarks from the literature. Besides, we provide insights on the practical relevance of the PLRP-PS on our instances. Compared to a classical CVRP model, we find that combining driving and walking reduces the driving time needed by 19% on average to deliver to all customers while also decreasing the overall working time by 30%. Most of this saving is achieved by reducing the overall parking time by 80%.

To complete this analysis, we find that the parking time has a major impact on solutions. Indeed, integrating a five-minute parking time reduces the average number of parking locations used in solutions by 59% compared to a situation where the parking time is null.

We conclude that integrating parking selection in heuristics can be efficiently done with simple approaches. In this context, the park-and-loop practice is of significant interest to reduce working and driving time in dense delivery areas. Interesting perspectives would be to investigate the impact of park-on-loop on $C0_2$ emissions as well as on the integration of temporal aspects such as time windows at customers.

Acknowledgment

We kindly thank Nicolás Cabrera and Jorge E. Mendoza for their insights on the PRLP instances, as well as Gwénaël Rault for his advice about the real-world data.

This research was supported by the Agence Nationale de la Recherche (ANR) under grant ANR-17-CE22-0015. This support is gratefully acknowledged.

References

Niels Agatz, Paul Bouman, and Marie Schmidt. Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, 52(4):965–981, 2018. doi:10.1287/trsc.2017.0791.

Oliver Bates, Bran Knowles, and Adrian Friday. Are people the key to enabling collaborative smart logistics? In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1494–1499, 2017. doi:10.1145/3027063.3053128.

Lawrence Bodin and Laurence Levy. Scheduling of local delivery carrier routes for the United States Postal Service. In Moshe Dror, editor, *Arc Routing: Theory, Solutions and Applications*, pages 419–442. Springer US, 2000. doi:10.1007/978-1-4615-4495-1_11.

Nicolás Cabrera, Jean-François Cordeau, and Jorge E Mendoza. The Doubly Open Park-and-loop Routing Problem. *Computers & operations research*, 2022. doi:10.1016/j.cor.2022.105761.

Giacomo Dalla Chiara, Lynette Cheah, Carlos Lima Azevedo, and Moshe E. Ben-Akiva. A Policy-Sensitive Model of Parking Choice for Commercial Vehicles in Urban Areas. *Transportation Science*, 54(3):606–630, 2020. doi:10.1287/trsc.2019.0970.

Jan Christiaens and Greet Vanden Berghe. Slack induction by string removals for vehicle routing problems. *Transportation Science*, 54(2):417–433, 2020. doi:10.1287/trsc.2019.0914.

Marc-Antoine Coindreau, Olivier Gallay, and Nicolas Zufferey. Vehicle routing with transportable resources: Using carpooling and walking for on-site services. *European Journal of Operational Research*, 279(3):996–1010, 2019. doi:10.1016/j.ejor.2019.06.039.

Teodor Gabriel Crainic, Nicoletta Ricciardi, and Giovanni Storchi. Models for Evaluating and Planning City Logistics Systems. *Transportation Science*, 43(4):432–454, 2009. doi:10.1287/trsc.1090.0279.

Rosario Cuda, Gianfranco Guastaroba, and M.Grazia Speranza. A survey on two-echelon routing problems. *Computers & Operations Research*, 55:185–199, 2015. doi:10.1016/j.cor.2014.06.008.

George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

Gunter Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational physics*, 104(1):86–92, 1993. doi:https://doi.org/10.1006/jcph.1993.1010.

Dorian Dumez. *Matheuristic approaches for solving transport optimization problems in urban logistics*. PhD thesis, IMT Atlantique, September 2021. URL `https://tel.archives-ouvertes.fr/tel-03394281`.

Dorian Dumez, Fabien Lehuédé, and Olivier Péton. A large neighborhood search approach to the vehicle routing problem with delivery options. *Transportation Research Part B: Methodological*, 144, 2021a. doi:10.1016/j.trb.2020.11.012.

Dorian Dumez, Christian Tilk, Stefan Irnich, Fabien Lehuédé, and Olivier Péton. Hybridizing Large Neighborhood Search and Exact Methods for Generalized Vehicle Routing Problems with Time Windows. *EURO Journal on Transportation and Logistics*, 2021b. doi:10.1016/j.ejtl.2021.100040.

Christian Fikar and Patrick Hirsch. A matheuristic for routing real-world home service transport systems facilitating walking. *Journal of Cleaner Production*, 105:300–310, 2015. doi:10.1016/j.jclepro.2014.07.013.

Gianfranco Guastaroba, M.Grazia Speranza, and Daniele Vigo. Intermediate Facilities in Freight Transportation Planning: A Survey. *Transportation Science*, 50(3):763–789, 2016. doi:10.1287/trsc.2015.0631.

Elisabeth Gussmagg-Pfliegl, Fabien Tricoire, Karl F. Doerner, and Richard F. Hartl. Multi-delivery problems with park-and-loop tours : a heuristic approach. In *Proceedings of OR Peripatetic Post-Graduate Programme (ORP3-2011) : Cádiz, Spain : September 13-17, 2011. - (Actas)*. Universidad de Cádiz, Cádiz, 2011. ISBN 978-84-9828-743-1. URL `http://digital.casalini.it/9788498287431`.

Vera C Hemmelmayr, Jean-François Cordeau, and Teodor Gabriel Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & operations research*, 49(12):3215–3228, 2012. doi:10.1016/j.cor.2012.04.007.

Stefan Irnich. Solution of real-world postman problems. *European journal of operational research*, 190(1):52–67, 2008.

Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983. doi:10.1126/science.220.4598.671.

Laurence Levy and Lawrence Bodin. The arc oriented location routing problem. *INFOR: Information Systems and Operational Research*, 27(1):74–94, 1989. doi:10.1080/03155986.1989.11732083.

Hongqi Li, Jun Chen, Feilong Wang, and Ming Bai. Ground-vehicle and unmanned-aerial-vehicle routing problems from two-echelon scheme perspective: a review. *European Journal of Operational Research*, 2021. doi:10.1016/j.ejor.2021.02.022.

Antonio Martinez-Sykora, Fraser McLeod, Carlos Lamas-Fernandez, Tolga Bektaş, Tom Cherrett, and Julian Allen. Optimised solutions to the last-mile delivery problem in London using a combination of walking and driving. *Annals of Operations Research*, 295(2):645–693, 2020. doi:10.1007/s10479-020-03781-8.

Sophie N. Parragh and Jean-François Cordeau. Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. *Computers & Operations Research*, 83:28–44, 2017. doi:10.1016/j.cor.2017.01.020.

David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8):2403–2435, 2007. doi:10.1016/j.cor.2005.09.012.

Sara Reed, Ann Melissa Campbell, and Barrett W. Thomas. Does Parking Matter? The Impact of Search Time for Parking on Last-Mile Delivery Optimization. *arXiv:2107.06788 [math]*, July 2021. URL `http://arxiv.org/abs/2107.06788`.

Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472, 2006a. doi:10.1287/trsc.1050.0135.

Stefan Ropke and David Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3):750–775, 2006b.

Gerhard Schrimpf, Johannes Schneider, Hermann Stamm-Wilbrandt, and Gunter Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171, 2000.

Frédéric Semet. A two-phase algorithm for the partial accessibility constrained vehicle routing problem. *Annals of Operations Research*, 61(1):45–65, 1995.

Gerald Senarclens de Grancy and Marc Reimann. Evaluating two new heuristics for constructing customer clusters in a VRPTW with multiple service workers. *Central European Journal of Operations Research*, 23(2):479–500, 2015. doi:10.1007/s10100-014-0373-4.

Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming*, pages 417–431. Springer, 1998.

Natasja Sluijk, Alexandre M Florio, Joris Kinable, Nico Dellaert, and Tom Van Woensel. Two-echelon vehicle routing problems: A literature review. *European Journal of Operational Research*, 2022.

Marius M. Solomon. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35(2):254–265, 1987. doi:10.1287/opre.35.2.254.

Appendix A. Computational experiments results details

Inst	VNS Coindreau et al. (2019)	MH Cabrera et al. (2022)	SLNS *this paper*
20_A_1	(2, 32.42)	**(2, 30.95)**	**(2, 30.95)**
20_A_2	(2, 41.62)	**(2, 41.56)**	**(2, 41.56)**
20_A_3	(2, 39.44)	**(2, 36.22)**	**(2, 36.22)**
20_A_4	(2, 39.16)	**(2, 36.03)**	**(2, 36.03)**
20_A_5	**(2, 35.27)**	**(2, 35.27)**	**(2, 35.27)**
20_A_6	(2, 43.72)	**(2, 42.25)**	**(2, 42.24)**
20_A_7	(2, 40.11)	(2, 38.67)	**(2, 38.63)**
20_A_8	(2, 39.15)	**(2, 36.85)**	**(2, 36.85)**
20_A_9	(2, 29.93)	**(2, 29.61)**	**(2, 29.61)**
20_A_10	(2, 41.23)	(2, 39.67)	**(2, 39.66)**
30_A_1	(3, 41.3)	**(3, 40.75)**	**(3, 40.75)**
30_A_2	(3, 46.65)	(3, 45.66)	**(3, 45.64)**
30_A_3	(3, 50.98)	**(3, 49.18)**	(3, 49.4)
30_A_4	(3, 46.32)	**(3, 43.76)**	**(3, 43.76)**
30_A_5	(3, 48.4)	**(3, 47.04)**	**(3, 47.04)**
30_A_6	(3, 51.63)	**(3, 49.58)**	**(3, 49.58)**
30_A_7	**(3, 45.53)**	**(3, 45.54)**	**(3, 45.53)**
30_A_8	(3, 45.36)	**(3, 43.69)**	(3, 43.93)
30_A_9	(3, 41.84)	**(3, 41.14)**	**(3, 41.14)**
30_A_10	(3, 49.53)	**(3, 46.88)**	(3, 47.16)
40_A_1	**(3, 59.17)**	**(3, 59.17)**	**(3, 59.17)**
40_A_2	**(3, 58.51)**	**(3, 58.51)**	(3, 59.66)
40_A_3	(3, 63.16)	**(3, 62.91)**	**(3, 62.91)**
40_A_4	**(3, 50.36)**	**(3, 50.36)**	**(3, 50.36)**
40_A_5	(4, 52.76)	**(4, 51.74)**	**(4, 51.74)**
40_A_6	(3, 63.08)	**(3, 61.27)**	**(3, 61.27)**
40_A_7	(3, 57.78)	(3, 54.9)	**(3, 54.82)**
40_A_8	(4, 56.37)	(4, 56.32)	**(4, 55.99)**
40_A_9	(3, 56.47)	**(3, 56.36)**	**(3, 56.36)**
40_A_10	(3, 57.7)	(3, 57.7)	**(3, 57.69)**
50_A_1	(4, 60.51)	**(4, 56.99)**	**(4, 56.99)**
50_A_2	(4, 62.49)	(4, 62.15)	**(4, 60.59)**
50_A_3	(4, 65.88)	(4, 63.87)	**(4, 63.67)**
50_A_4	(4, 58.22)	(4, 57.93)	**(4, 56.94)**
50_A_5	(4, 64.18)	**(4, 64.12)**	(4, 64.96)
50_A_6	(4, 66.14)	**(4, 65.03)**	(4, 65.35)
50_A_7	**(4, 63.71)**	**(4, 63.71)**	(4, 63.82)
50_A_8	(4, 70.89)	(4, 69.78)	**(4, 68.82)**
50_A_9	**(4, 58.2)**	(4, 58.73)	**(4, 58.2)**
50_A_10	**(4, 60.79)**	(4, 61.23)	**(4, 60.78)**

Table A.6: Comparison with the VNS of Coindreau et al. (2019) and the matheuristic of Cabrera et al. (2022) (Coindreau's instances)

Instance	filter-r5	filter-r10	filter-r15	filter-k5	filter-k10	filter-k15	sort-r5	sort-r10	sort-r15	sort-k3	sort-k5	sort-k10	no filter
nantes-50-1	(3, 394.13)	(3, 364.97)	(3, 329.83)	(3, 328.97)	**(3, 326.5)**	(3, 329.4)	(3, 330.65)	(3, 330.45)	(3, 329.83)	(3, 330.27)	(3, 330.45)	(3, 329.08)	(3, 329.35)
nantes-50-2	(4, 462.58)	(4, 466.15)	**(3, 362.6)**	(3, 363.42)	(3, 364.53)	(3, 365.25)	(3, 363.08)	(3, 366.15)	(3, 363.72)	(3, 363.05)	(3, 363.73)	(3, 363.08)	(3, 363.4)
nantes-50-3	(4, 423.97)	(4, 390.85)	(4, 307.35)	(4, 306.88)	(4, 306.12)	(4, 307.32)	(4, 306.55)	(4, 305.95)	(4, 307.65)	(4, 307.23)	**(4, 305.73)**	(4, 307.48)	(4, 306.88)
nantes-50-4	(4, 397.18)	(4, 367.58)	(4, 324.9)	(4, 324.72)	**(4, 323.78)**	(4, 323.98)	(4, 324.32)	(4, 326.23)	(4, 325.62)	(4, 323.98)	(4, 324.55)	(4, 324.35)	(4, 324.83)
nantes-50-5	(3, 393.63)	(3, 365.82)	(3, 324.42)	(3, 324.62)	(3, 324.73)	(3, 324.37)	(3, 324.28)	**(3, 323.87)**	(3, 324.02)	(3, 324.62)	(3, 324.98)	(3, 324.1)	(3, 324.58)
nantes-50-6	(4, 365.07)	(4, 360.85)	(4, 339.98)	(4, 340.07)	(4, 340.1)	(4, 340.03)	(4, 340.22)	(4, 337.75)	(4, 340.82)	(4, 339.32)	(4, 339.25)	**(4, 337.72)**	(4, 339.13)
nantes-50-7	(4, 423.83)	(4, 413.93)	(4, 343.6)	**(4, 342.95)**	(4, 344.88)	(4, 343.2)	(4, 343.65)	(4, 344.93)	(4, 343.38)	(4, 344.22)	(4, 343.92)	(4, 342.95)	(4, 343.93)
nantes-50-8	(4, 429.28)	(4, 403.98)	(4, 335.15)	**(4, 333.72)**	(4, 334.23)	(4, 335.25)	(4, 334.32)	(4, 336.93)	(4, 335.47)	(4, 337.07)	(4, 333.75)	(4, 334.53)	(4, 334.52)
nantes-50-9	(4, 413.73)	(4, 402.15)	(4, 343.7)	(4, 341.62)	(4, 341.7)	(4, 342.45)	(4, 341.98)	(4, 344.27)	(4, 343.63)	**(4, 341.55)**	(4, 343.18)	(4, 341.57)	(4, 343.92)
nantes-50-10	(4, 421.1)	(4, 406.82)	(4, 345.8)	(4, 344.93)	(4, 348.83)	(4, 347.75)	(4, 345.38)	**(4, 344.18)**	(4, 346.58)	(4, 344.85)	(4, 346.37)	(4, 345.65)	(4, 347.22)
nantes-100-1	(7, 716.07)	(7, 701.25)	(7, 618.18)	(7, 619.42)	**(7, 604.97)**	(7, 610.1)	(7, 621.0)	(7, 610.43)	(7, 628.93)	(7, 611.4)	(7, 615.47)	(7, 613.65)	(7, 615.63)
nantes-100-2	(7, 655.6)	(7, 643.8)	(7, 579.77)	(7, 579.95)	(7, 576.83)	(7, 580.52)	(7, 576.62)	(7, 576.65)	(7, 582.43)	(7, 577.37)	(7, 578.13)	(7, 575.03)	**(7, 572.72)**
nantes-100-3	(6, 658.38)	(6, 628.42)	(6, 533.52)	(6, 530.25)	(6, 530.55)	**(6, 525.7)**	(6, 527.92)	(6, 531.25)	(6, 532.98)	(6, 526.43)	(6, 527.18)	(6, 532.12)	(6, 530.63)
nantes-100-4	(6, 683.22)	(6, 623.9)	(6, 531.95)	(6, 529.78)	(6, 530.27)	**(6, 524.63)**	(6, 531.97)	(6, 528.73)	(6, 528.55)	(6, 524.9)	(6, 525.27)	(6, 529.17)	(6, 529.68)
nantes-100-5	(7, 662.05)	(7, 640.22)	(7, 557.35)	(7, 554.28)	(7, 559.87)	(7, 554.17)	**(7, 551.5)**	(7, 557.67)	(7, 558.6)	(7, 556.35)	(7, 558.43)	(7, 560.68)	(7, 555.0)
nantes-100-6	(7, 686.48)	(7, 671.5)	(7, 588.95)	(7, 580.12)	**(7, 579.23)**	(7, 582.92)	(7, 582.28)	(7, 585.8)	(7, 584.43)	(7, 582.62)	(7, 583.5)	(7, 587.33)	(7, 582.7)
nantes-100-7	(7, 702.68)	(7, 700.88)	(7, 619.55)	(7, 617.02)	**(7, 613.4)**	(7, 617.38)	(7, 617.3)	(7, 621.7)	(7, 625.27)	(7, 614.25)	(7, 615.9)	(7, 615.28)	(7, 616.18)
nantes-100-8	(6, 611.92)	(6, 589.85)	(6, 525.98)	(6, 527.13)	(6, 530.08)	(6, 530.85)	(6, 529.5)	(6, 534.82)	(6, 530.23)	(6, 532.82)	(6, 529.05)	(6, 530.17)	**(6, 523.97)**
nantes-100-9	(7, 741.35)	(7, 715.7)	(7, 640.07)	(7, 633.95)	(7, 638.17)	(7, 634.38)	(7, 644.07)	(7, 633.22)	(7, 635.92)	(7, 634.68)	(7, 631.42)	**(7, 629.47)**	(7, 634.17)
nantes-100-10	(7, 750.45)	(7, 704.55)	(7, 621.38)	(7, 621.0)	(7, 619.9)	(7, 621.32)	(7, 626.65)	**(7, 619.75)**	(7, 620.53)	(7, 622.57)	(7, 622.23)	(7, 625.0)	(7, 624.68)
nantes-200-1	(13, 1166.72)	(13, 1096.33)	(13, 1015.5)	(13, 994.3)	(13, 999.38)	(13, 1011.32)	(13, 1003.48)	(13, 1017.07)	(13, 1007.73)	(13, 989.22)	(13, 993.8)	(13, 997.65)	**(13, 984.42)**
nantes-200-2	(12, 1221.25)	(12, 1143.72)	(12, 1024.23)	(12, 1006.42)	(12, 1003.43)	(12, 993.67)	(12, 1009.37)	(12, 1007.25)	(12, 1010.52)	(12, 1002.42)	**(12, 989.85)**	(12, 1003.32)	(12, 1012.12)
nantes-200-3	(12, 1182.07)	(12, 1126.3)	(12, 1020.77)	(12, 1000.47)	**(12, 978.95)**	(12, 999.38)	(12, 1015.95)	(12, 997.27)	(12, 997.68)	(12, 991.92)	(12, 998.75)	(12, 985.85)	(12, 993.87)
nantes-200-4	(12, 1067.12)	(12, 1071.67)	(12, 984.5)	(12, 977.88)	(12, 977.57)	(12, 986.95)	(12, 995.92)	(12, 989.03)	(12, 997.43)	**(12, 968.28)**	(12, 976.77)	(12, 987.97)	(12, 995.1)
nantes-200-5	(12, 1094.37)	(12, 1092.02)	(12, 991.13)	(12, 998.15)	(12, 994.5)	(12, 1000.63)	(12, 1007.22)	(12, 1015.18)	(12, 997.03)	**(12, 976.8)**	(12, 987.5)	(12, 993.47)	(12, 991.78)
nantes-200-6	(12, 1081.63)	(12, 1087.07)	(12, 991.4)	(12, 957.02)	(12, 967.5)	(12, 978.33)	(12, 975.43)	(12, 990.98)	(12, 989.45)	(12, 958.97)	**(12, 940.2)**	(12, 966.95)	(12, 968.93)
nantes-200-7	(12, 1126.8)	(12, 1077.27)	(12, 990.83)	(12, 986.55)	(12, 980.43)	(12, 996.12)	(12, 978.4)	(12, 1012.07)	(12, 993.63)	(12, 981.23)	(12, 970.15)	(12, 980.52)	**(12, 968.8)**
nantes-200-8	(13, 1116.5)	(13, 1051.87)	(13, 938.15)	(13, 934.18)	(13, 940.7)	(13, 932.62)	(13, 928.63)	(13, 943.7)	(13, 941.58)	(13, 923.9)	(13, 936.35)	**(13, 921.73)**	(13, 938.38)
nantes-200-9	(13, 1109.72)	(13, 1052.28)	(13, 981.4)	(13, 971.88)	(13, 969.87)	(13, 976.13)	(13, 992.15)	(13, 985.47)	(13, 989.02)	**(13, 967.87)**	(13, 973.72)	(13, 970.33)	(13, 972.65)
nantes-200-10	(12, 1123.42)	(12, 1083.02)	(12, 1003.97)	(12, 978.9)	**(12, 968.48)**	(12, 971.12)	(12, 997.38)	(12, 996.72)	(12, 994.4)	(12, 989.82)	(12, 983.85)	(12, 984.73)	(12, 990.05)
nantes-300-1	(18, 1625.17)	(18, 1637.25)	(18, 1420.27)	(18, 1384.92)	**(18, 1378.35)**	(18, 1398.98)	(18, 1409.67)	(18, 1442.97)	(18, 1450.47)	(18, 1407.18)	(18, 1389.55)	(18, 1383.2)	(18, 1402.13)
nantes-300-2	(18, 1487.53)	(18, 1489.1)	(18, 1381.03)	(18, 1337.13)	(18, 1358.18)	(18, 1348.87)	(18, 1378.48)	(18, 1361.38)	(18, 1378.52)	**(18, 1331.9)**	(18, 1338.57)	(18, 1341.0)	(18, 1339.2)
nantes-300-3	(19, 1618.9)	(19, 1560.13)	(19, 1396.48)	(19, 1376.3)	**(19, 1369.07)**	(19, 1386.07)	(19, 1404.98)	(19, 1427.65)	(19, 1402.15)	(19, 1373.37)	(19, 1383.88)	(19, 1376.72)	(19, 1400.75)
nantes-300-4	(18, 1656.42)	(18, 1633.35)	(18, 1400.62)	(18, 1401.47)	(18, 1379.23)	(18, 1414.65)	(18, 1397.32)	(18, 1408.3)	(18, 1452.85)	(18, 1416.75)	(18, 1401.82)	**(18, 1370.55)**	(18, 1385.2)
nantes-300-5	(19, 1559.87)	(19, 1524.55)	(19, 1339.23)	(19, 1341.6)	**(19, 1322.23)**	(19, 1335.07)	(19, 1347.22)	(19, 1374.43)	(19, 1348.68)	(19, 1343.12)	(19, 1341.38)	(19, 1340.37)	(19, 1347.92)
nantes-300-6	(18, 1667.8)	(18, 1577.6)	(18, 1399.07)	(18, 1380.77)	(18, 1385.08)	(18, 1374.3)	(18, 1434.87)	(18, 1396.4)	(18, 1414.38)	(18, 1371.25)	**(18, 1361.97)**	(18, 1366.3)	(18, 1371.17)
nantes-300-7	(18, 1541.17)	(18, 1530.37)	(18, 1308.28)	(18, 1295.32)	(18, 1285.2)	(18, 1293.52)	(18, 1301.9)	(18, 1313.13)	(18, 1322.07)	(18, 1282.25)	**(18, 1261.27)**	(18, 1266.67)	(18, 1287.12)
nantes-300-8	(18, 1568.02)	(18, 1698.68)	(18, 1394.57)	(18, 1377.95)	(18, 1386.05)	(18, 1423.38)	(18, 1412.95)	(18, 1419.5)	(18, 1472.08)	(18, 1389.57)	**(18, 1367.65)**	(18, 1380.85)	(18, 1392.12)
nantes-300-9	(19, 1570.82)	(19, 1484.25)	(19, 1369.03)	**(19, 1349.72)**	(19, 1364.68)	(19, 1364.65)	(19, 1373.45)	(19, 1374.73)	(19, 1386.53)	(19, 1361.72)	(19, 1356.28)	(19, 1359.73)	(19, 1386.53)
nantes-300-10	(19, 1596.87)	(19, 1579.23)	(19, 1417.85)	(19, 1378.28)	(19, 1395.42)	(19, 1384.38)	(19, 1407.3)	(19, 1418.43)	(19, 1425.57)	(19, 1379.4)	(19, 1375.97)	(19, 1399.62)	**(19, 1373.07)**
nantes-400-1	(25, 2061.32)	(25, 2050.38)	(25, 1785.22)	(25, 1774.08)	**(25, 1766.75)**	(25, 1811.77)	(25, 1771.32)	(25, 1807.92)	(25, 1809.03)	(25, 1785.57)	(25, 1771.83)	(25, 1811.85)	(25, 1814.4)
nantes-400-2	(24, 2022.18)	(24, 1951.85)	(24, 1743.28)	(24, 1690.55)	(24, 1701.23)	(24, 1710.03)	(24, 1757.77)	(24, 1734.48)	(24, 1772.07)	(24, 1723.28)	(24, 1713.9)	**(24, 1686.92)**	(24, 1734.43)
nantes-400-3	(25, 1943.32)	(25, 1971.98)	(25, 1706.42)	(25, 1707.93)	**(25, 1650.48)**	(25, 1707.3)	(25, 1696.75)	(25, 1697.4)	(25, 1742.3)	(25, 1692.95)	(25, 1686.45)	(25, 1680.73)	(25, 1693.4)
nantes-400-4	(25, 2008.33)	(25, 1990.68)	(25, 1723.57)	(25, 1682.8)	(25, 1687.45)	**(25, 1677.05)**	(25, 1724.35)	(25, 1724.52)	(25, 1737.23)	(25, 1690.27)	(25, 1693.88)	(25, 1710.68)	(25, 1706.22)
nantes-400-5	(24, 2121.28)	(24, 2090.17)	(24, 1804.73)	**(24, 1778.63)**	(24, 1803.63)	(24, 1805.47)	(24, 1812.75)	(24, 1829.4)	(24, 1844.47)	(24, 1783.32)	(24, 1806.4)	(24, 1800.18)	(24, 1797.43)
nantes-400-6	(24, 1933.15)	(24, 1931.48)	(24, 1751.58)	(24, 1734.1)	**(24, 1716.92)**	(24, 1729.7)	(24, 1784.22)	(24, 1764.17)	(24, 1801.03)	(24, 1723.18)	(24, 1757.55)	(24, 1732.95)	(24, 1736.43)
nantes-400-7	(24, 1966.78)	(24, 2013.65)	(24, 1711.0)	(24, 1692.52)	(24, 1717.43)	(24, 1737.25)	(24, 1761.03)	(24, 1711.5)	(24, 1768.63)	(24, 1702.48)	(24, 1714.48)	**(24, 1685.37)**	(24, 1737.9)
nantes-400-8	(25, 2017.52)	(25, 2015.9)	(25, 1767.53)	(25, 1719.48)	(25, 1729.77)	(25, 1755.25)	(25, 1775.63)	(25, 1777.77)	(25, 1787.73)	**(25, 1713.75)**	(25, 1725.78)	(25, 1715.77)	(25, 1751.38)
nantes-400-9	(24, 1989.23)	(24, 2021.07)	(24, 1690.33)	(24, 1659.85)	(24, 1705.75)	(24, 1673.42)	(24, 1692.43)	(24, 1711.08)	(24, 1723.52)	**(24, 1659.27)**	(24, 1668.98)	(24, 1674.32)	(24, 1688.73)
nantes-400-10	(25, 2009.97)	(25, 1879.88)	(25, 1710.65)	(25, 1679.7)	(25, 1691.43)	(25, 1700.32)	(25, 1695.63)	(25, 1684.68)	(25, 1698.0)	**(25, 1646.53)**	(25, 1655.9)	(25, 1656.2)	(25, 1667.6)

Table A.7: Detailed comparison of parking selection strategies. For each strategy and each instance, the table present the number of vehicles and overall working time.