Received: 6 January 2021 | Accepted: 10 January 2021 | Published on: 27 February 2021

DOI: 10.1002/net.22028

RESEARCH ARTICLE

WILEY

A solution approach for multi-trip vehicle routing problems with time windows, fleet sizing, and depot location

Paula Fermín Cueto | **Ivona Gjeroska** | **Albert Solà Vilalta** | **Miguel F. Anjos**

Maxwell Institute for Mathematical Sciences, School of Mathematics, University of Edinburgh, Edinburgh, United Kingdom

Correspondence
Miguel F. Anjos, Maxwell Institute for Mathematical Sciences, School of Mathematics, University of Edinburgh, Edinburgh, United Kingdom.
Email: anjos@stanfordalumni.org

Funding information
This research was supported by the Centre for Doctoral Training funded by the UK Engineering and Physical Sciences Research Council, Grant/Award Number: EP/L016508/01. The Maxwell Institute Graduate School in Analysis and its Applications. Scottish Funding Council. Heriot-Watt University. University of Edinburgh.

Abstract

We present a solution approach for a multi-trip vehicle routing problem with time windows in which the locations of a prescribed number of depots and the fleet sizes must also be optimized. Given the complexity of the task, we divide the problem into subproblems that are solved sequentially. First, we address strategic decisions, which are solved once and remain constant thereafter. Depots are allocated by solving a *p*-median problem and fleet sizes are determined by identifying the vehicle requirements of several worst-case demand instances. Then, we address the operational planning aspect: optimizing the vehicle routes on a daily basis to satisfy the fluctuating customer demand. We assign customers to depots based on distance and "routing effort," and for the routing problem we combine a tailor-made branch-and-cut algorithm with a heuristic consisting of a route construction phase and packing of routes into vehicle trips. Our strategic decision models are robust in the sense that when applied to unseen data, all customers could be visited with the allocated fleet sizes and depot locations. Our operational routing methods are both time and cost-effective. The exact method yields acceptable optimality gaps in 20 min and the heuristic runs in less than 2 min, finding optimal or near-optimal solutions for small instances. Finally, we explore the trade-off between depot and fleet costs, and routing costs to make recommendations on the optimal number of depots. Our solution approach was entered into the 12th AIMMS-MOPTA Optimization Modeling Competition and was awarded the first prize.

KEYWORDS

branch-and-cut, fleet sizing, heuristics, multiple depots, multiple trips, subtour elimination constraints, time windows, vehicle routing

1 | INTRODUCTION

The vehicle routing problem (VRP) is among the most studied problems in the field of combinatorial optimization. It is a generalization of the well-known traveling salesman problem (TSP) where instead of a single salesman there is a fixed number of identical vehicles leaving from and returning to a given depot. The VRP has many practical applications and it can be tailored to model a wide variety of business-specific needs. Many of these special cases can be found in the literature, and a variety of exact methods and heuristic approaches have been developed to obtain good solutions in a reasonable amount of time [17]. Just like the TSP, the VRP is an intriguing problem in that it is simple to explain and yet extremely challenging to solve.

Formally, the VRP can be defined as follows: let $G = (V, E)$ be a connected, undirected graph, where V is the set of nodes and E is the set of edges. Each node represents a customer and each edge is associated with the traveling time (or distance) between two customers. There is a fleet of $|K|$ identical vehicles of capacity Q. Each customer $i \in V$ has a fixed demand of q_i, and all of its demand must be satisfied by a single vehicle. The objective is to construct a single route for each vehicle such that the total distance traveled is minimized. This is commonly known as the capacitated VRP (CVRP), which is the basis of all further variations of the problem. A more detailed description of it can be found in [22] and a good overview of the work that has been done in the past years on the VRP is given in a survey paper by Laporte [17].

The modeling problem proposed for the 12th AIMMS-MOPTA Optimization Modeling Competition [1] was a variant of the CVRP with some additional elements. (i) It is required that every customer be visited within a given time window (8:00 a.m. and 4:00 p.m.) and, additionally, vehicles can only operate within fixed time intervals (from 6:00 a.m. to 5:00 p.m.). This variant can be found in the literature as the *VRP with Time Windows (VRPTW)* [15]. (ii) In this problem, there is not a single depot; a fixed number of depots is given as a problem parameter; the locations of these depots must be determined. These are also well studied problems and they are referred to in the literature as the *multi-depot VRP* and the *location routing problem*, respectively. Laporte et al. describe an algorithm for solving a combination of these two problems efficiently in [18]. (iii) The size of the fleet assigned to each depot must be minimized and this assignment must remain constant for the entire duration of the problem. A problem of this type was first introduced by Golden et al. in [13] and significant amount of work has been done on this topic since then. Baldaci et al. introduce in [3] a new mathematical formulation for a problem of this type with a heterogeneous fleet—a generalization of our fleet sizing problem, as we only need to consider a homogeneous fleet. Lastly, (iv) vehicles are allowed to perform multiple trips, starting and terminating at the same depot. This is often referred to as the *Multi-Trip VRP (MTVRP)* [6]. We use the term *trip* in the context of the multi-trip aspect of the problem: it refers to an occasion where a given vehicle leaves the depot to visit customers and return. By contrast, we use the term *route* to refer to the actual path that a vehicle follows during a trip.

All of the problems listed above are known to be computationally challenging, and heuristic methods have been developed as a consequence of the limitations of the exact approaches.

The data provided for this problem as part of the 12th AIMMS-MOPTA Optimization Modeling Competition [1] includes a graph of 1457 nodes representing ZIP codes in the state of Pennsylvania, USA. The given graph is not complete, but it is connected, so a path can be found between any two nodes using existing edges.

Demand of each customer is provided for every day in 2018 and 2019. Despite the large number of customers, on average only 5% must be visited on a given day. The 2018 data is used for model development while the 2019 data is reserved for testing the robustness of our solution method. In practice, demand for a given day is not known until 6:00 p.m. the day before, so all routes should be obtained quickly enough to inform the drivers on time.

We propose a method that efficiently determines the location of a given number of depots, optimizes the fleet size and produces a sensible routing strategy in a reasonable time.

This article is structured as follows: in Section 2 we present a mathematical formulation of the problem. Our methodology is outlined in Section 3. Sections 4 and 5 present an analysis of the minimum number of depots and the depot location model, respectively. In Section 6, a robust approach to the fleet sizing problem is proposed. Section 7 describes an algorithm to assign customers to depots on a daily basis. Both exact and heuristic approaches for the routing problem are described in Section 8. Solutions are presented in Section 9. We present a methodology for automating the choice of the number of depots in Section 10 and lastly, conclusions and recommendations are given in Section 11.

2 | MATHEMATICAL FORMULATION

In this section, we present a mixed integer linear programming formulation for the multi-trip VRP with time windows, fleet sizing and depot location. Some assumptions were made at different stages of the problem:

- Each customer must be served by exactly one vehicle.
- The demand of customers colocated with a depot is assumed to be served automatically.
- The distance between two customers is the shortest path between their locations through known routes. The Dijkstra algorithm [9] was applied to generate a distance matrix with all the shortest paths between any two nodes in the network.
- No time is needed to unload the demand at a customer location or to reload at the depot.

We define an undirected graph $G = (V, E)$ where the set of nodes V represents the customers and the set of edges E represents the shortest paths between customers. We also define a set of unknown depot locations, a set of potential vehicles to be used and a set of trips each vehicle might take. Formally, we introduce the following mathematical formulation:

Sets:

V	Set of nodes
E	Set of edges, that is, $E = V \times V$
$H \subseteq V$	Set of depots

K	Set of vehicles
R	Set of trips per vehicle

Parameters:

f_i	Cost of setting a depot on location $i \in V$
c_{ij}	Cost of traversing edge $(i,j) \in E$
u	Cost of a vehicle
Q	Capacity of a vehicle
q_i	Demand at node $i \in V$
t_{ij}	Time required to travel from $i \in V$ to $j \in V$
$[a^c, b^c]$	Time window of customers
$[a^d, b^d]$	Depot opening hours

Decision variables:

$$x_{ij}^{kr} = \begin{cases} 1 & \text{if vehicle } k \text{ traverses edge } (i,j) \text{ in trip } r, \\ 0 & \text{otherwise.} \end{cases}$$

$$z_i = \begin{cases} 1 & \text{if there is a depot at node } i, \\ 0 & \text{otherwise.} \end{cases}$$

$$w_i^k = \begin{cases} 1 & \text{if vehicle } k \text{ is assigned to a depot} \\ & \text{located at node } i, \\ 0 & \text{otherwise.} \end{cases}$$

s_i	Time node i is visited (if i is not a depot)
h_1^{kr}	Time the depot is visited by its assigned vehicle k at the *start* of trip r
h_2^{kr}	Time the depot is visited by its assigned vehicle k at the *end* of trip r

Minimize

$$\sum_{i\in V}\sum_{j\in V} c_{ij} \sum_{k\in K}\sum_{r\in R} x_{ij}^{kr} + \sum_{i\in V} f_i z_i + u \sum_{k\in K}\sum_{i\in V} w_i^k \tag{1}$$

subject to

$$\sum_{j\in V}\sum_{k\in K}\sum_{r\in R} x_{ji}^{kr} \geq 1 \quad \forall\, i \in V \tag{2}$$

$$\sum_{j\in V}\sum_{k\in K}\sum_{r\in R} x_{ji}^{kr} \leq 1 + |V|\, z_i \quad \forall\, i \in V \tag{3}$$

$$\sum_{j\in V} x_{ij}^{kr} = \sum_{j\in V} x_{ji}^{kr} \quad \forall\, i \in V,\ k \in K,\ r \in R \tag{4}$$

$$\sum_{j\in V} x_{ij}^{kr} \leq 1 \quad \forall\, i \in V,\ k \in K,\ r \in R \tag{5}$$

$$\sum_{i\in V} q_i \left(\sum_{j\in V} x_{ij}^{kr} - w_i^k \right) \leq Q \quad \forall\, k \in K,\ r \in R \tag{6}$$

$$\sum_{i\in V} z_i = |H| \tag{7}$$

$$w_i^k \leq z_i \quad \forall\, i \in V,\ k \in K \tag{8}$$

$$\sum_{i\in V} w_i^k \leq 1 \quad \forall\, k \in K \tag{9}$$

$$x_{ij}^{kr} \leq \sum_{\ell\in V} w_\ell^k \quad \forall\, i,j \in V,\ k \in K,\ r \in R \tag{10}$$

$$\sum_{j\in V}\sum_{r\in R} x_{ij}^{kr} \geq w_i^k \quad \forall\, i \in V,\ k \in K \tag{11}$$

$$\sum_{\ell\in V}\sum_{r\in R} x_{\ell j}^{kr} \leq |V|\,(2 - w_i^k - z_j) \quad \forall\, i,j \in V,\ j \neq i, k \in K \tag{12}$$

$$\sum_{i\in V} w_i^{k+1} \le \sum_{i\in V} w_i^k \quad \forall\, k \in \{1, \ldots, |K|-1\} \tag{13}$$

$$\sum_{j\in V} x_{ij}^{k,r+1} \le \sum_{j\in V} x_{ij}^{kr} + 1 - z_i \quad \forall\, i \in V,\ k \in K,\ r \in \{1, \ldots, |R|-1\} \tag{14}$$

$$s_i + t_{ij} \le s_j + 2\, b^d (1 - x_{ij}^{kr} + z_j + z_i) \quad \forall\, i,j \in V,\ k \in K,\ r \in R \tag{15}$$

$$h_1^{kr} + t_{ij} \le s_j + 2\, b^d (2 - x_{ij}^{kr} + z_j - z_i) \quad \forall\, i,j \in V,\ k \in K,\ r \in R \tag{16}$$

$$s_i + t_{ij} \le h_2^{kr} + 2\, b^d (2 - x_{ij}^{kr} - z_j + z_i) \quad \forall\, i,j \in V,\ k \in K,\ r \in R \tag{17}$$

$$h_2^{kr} \le h_1^{k,r+1} \quad \forall\, k \in K,\ r \in \{1, \ldots, |R|-1\} \tag{18}$$

$$a^c \le s_i \le b^c \quad \forall\, i \in V \tag{19}$$

$$a^d \le h_1^{kr} \quad \forall\, k \in K,\ r \in R \tag{20}$$

$$h_2^{kr} \le b^d \quad \forall\, k \in K,\ r \in R \tag{21}$$

$$x_{ij}^{kr},\ z_i,\ w_i^k \in \{0, 1\} \tag{22}$$

$$s_i,\ h_1^{kr},\ h_2^{kr} \in \mathbb{R} \tag{23}$$

The objective function minimizes three types of costs: the variable routing costs (fuel, vehicle maintenance, tolls, etc.), the fixed depot costs (depot lease/purchase and personnel wages), and the fixed vehicle costs (lease/purchase, insurance, driver wages, etc.).

Constraints (2) and (3) ensure that every node $i \in V$ is visited *exactly once* if it is not a depot and *at least once* otherwise. Constraints (4) stipulate that, if trip r of vehicle k enters node i, it must leave node i. We introduce constraints (5) to ensure that every vehicle leaves a depot at most once in each trip; they are necessary to avoid sub-trips within the same trip that could deceive the time window constraints by taking place simultaneously. The vehicle capacity constraints are defined in (6). If node i is the depot of vehicle k, then the demand of this node is excluded from the load. Constraints (7) ensure that exactly $|H|$ depots are placed. Constraints (8)–(9) assign vehicles to at most one depot. Constraints (10) ensure that if a vehicle is not used, this vehicle cannot traverse any arcs. Constraints (11) impose that a vehicle that does not complete any trip cannot be marked as used by variables w. This is needed to prevent negative demand on a node when evaluating the capacity constraints (6), which could happen if it is not costly to mark a vehicle as used.

We introduce constraints (12) to ensure that every vehicle in use always departs from its assigned depot by forbidding it to enter other depots. We also introduce symmetry breaking constraints (13) and (14) to impose a natural ordering in the selected vehicles and trips, respectively, and eliminate equivalent solutions. They ensure that vehicle $k+1$ cannot be used unless k is in use; the same is true for trips $r+1$ and r.

Constraints (15)–(21) are our adaptation of the time window constraints for the multi-depot multi-trip VRP. (15) are used when none of the nodes considered are a depot. (16) only applies when the first node is a depot and (17), when the last node is a depot. Additional variables h_1^{kr}, h_2^{kr} are needed to define visit times to the depot, as there are two times associated with the depot for every vehicle and trip: start and end time of a trip. Because of this, we do not need the vehicle index nor the trip index in s_i, as nodes that are not depots can only be visited once by one vehicle. Constraints (18) ensure that a trip r cannot start before the finish time of the previous trip completed by the same vehicle. Finally, Constraints (19)–(21) enforce the time windows of customers and vehicles. Subtour elimination constraints are not necessary due to the presence of the time window constraints [15].

Using the formulation above, we can prove the complexity of the problem. By fixing the number of trips to one, the index r of any decision variables can be dropped. We can fix the depot at node 0 by forcing $z_0 = 1$, $|H| = 1$, and setting $a^d = a^c$ and $b^d = b^c$. With these changes, we notice that by removing the resulting redundant constraints, we obtain the VRPTW formulation. This shows that our problem is a *generalization* of the VRPTW. Since the VRPTW is itself a generalization of the TSP, which belongs to the class of *NP-hard* problems, we conclude that our problem is also *NP-hard* - there is no known algorithm that can solve this problem in polynomial time.

3 | METHODOLOGY

In this problem, there is a hierarchy of decisions that take place at different stages in the planning period:

1. *Strategic level*. Decisions involving capital expenditure (capex) that need to be made in advance and remain constant throughout the entire planning horizon; these are the location of depots and the fleet sizing.
2. *Operational level*. Decisions that are made on a daily basis and have an impact on operational expenditure (opex); these are the vehicle routes. Since customer demand is only known at 6:00 p.m. the day before, these routes should be generated fast enough so that schedules can be communicated to drivers before they start their shift at 6:00 a.m.

Because of this hierarchy, there is a business need to decompose the problem into separate subproblems. We have developed a solution approach that addresses the different stages of the problem sequentially. First, the location of depots is optimized. Next, fleet sizes are determined for each depot. After these strategic decisions have been fixed, we address the operational planning aspect: on a given day, customers that require a visit are assigned to depots and vehicle routes are optimized.

All our solution methods are implemented in AIMMS, with CPLEX as the MIP solver.

4 | MINIMUM NUMBER OF DEPOTS

Since the number of depots is a parameter and not a decision variable, we are interested in identifying the minimum number of depots that preserves feasibility. This number can be determined by solving a set covering problem (SCP). The SCP aims to place facilities at locations such that every node can be reached from at least one facility. Each depot can be considered to have a radius of coverage equal to the longest distance that a vehicle could cover in a single trip without violating the time windows.

In our problem, the radius of coverage is $d_{max} = 5.5 \text{ h} \times 40 \text{ mph} = 220$ miles. A vehicle could visit a customer located at most 5.5 h away from the depot, although this would likely result in a large fleet size with a poor vehicle utilization. Given the data, the solution to the SCP is trivial, as there are 10 ZIP codes such that the location of all customers is less than 5.5 h away from any of these 10 locations. Therefore, one depot suffices to satisfy all the demand in 2018, provided there are enough vehicles to complete the generated trips.

5 | DEPOT LOCATION

To optimize the location of a given number of depots without solving the routings for every day in the 2018 data, which would require very costly computations, we treat this as a facility location problem. We are interested in finding the best locations of a given number of depots such that if all customers are assigned to their closest depot, the weighted average distance between customers and their assigned depots will be minimum. The rationale is that minimizing this metric should result in shorter vehicle trips.

To obtain the weights for the average distance, we aggregate demand in 2018 by the number of days that each customer has positive demand. The more often a customer requires a visit, the more it will drive the location of the nearest depot toward it. We observe a strong correlation between this metric and the total annual demand of a customer in our data (the Pearson coefficient between the two metrics is $\rho = 0.998$). Therefore, choosing one metric or the other is not expected to have a significant impact on the outcome of the model.

In addition, the fixed costs of depots are also included in the objective function. Since we cannot estimate routing and vehicle costs from the average customer-depot distances, it is not straightforward to find the right balance between the depot costs and the average customer-depot distance. To overcome this difficulty, we include both terms in the objective function with a parameter $\alpha \in [0, 1]$ that controls their relative importance and we choose the best value for α by inspection.

Our formulation is a variation of the p-median problem [8]. Variables z_i indicate whether a depot is placed at location $i \in V$ and variables y_{ij} indicate whether node i is assigned to a depot located in j. d_{ij} is the distance between nodes i and j, $d_{\max}$ is the maximum distance allowed between a demand node and a depot due to the time window constraints, and Ω_i is the number of days that customer $i \in V$ had positive demand in 2018, namely $\Omega_i = \sum_d \mathbb{1}_{(q_{id}>0)}$.

Minimize

$$\frac{\alpha}{\sum_{i\in V}\Omega_i}\sum_{i\in V}\sum_{j\in V}\Omega_i \, y_{ij} \, d_{ij} + (1-\alpha)\sum_{j\in V} f_j \, z_j \tag{24}$$

subject to

$$\sum_{j\in V} y_{ij} = 1 \quad \forall\, i \in V, \tag{25}$$

$$\sum_{j\in V} z_j = |H|, \tag{26}$$

$$y_{ij} \le z_j \quad \forall\, i,j \in V, \tag{27}$$

$$d_{ij}\, y_{ij} \le d_{\max}\, z_j \quad \forall\, i,j \in V, \tag{28}$$

$$y_{ij},\ z_j \in \{0,1\} \quad \forall\, i,j \in V. \tag{29}$$

Constraints (25) ensure each node is assigned to exactly one depot. (26) is the p-median constraint, where the p parameter is represented by the cardinality of the set of depots, $|H|$. Constraints (27) stipulate that nodes can only be assigned to depots,

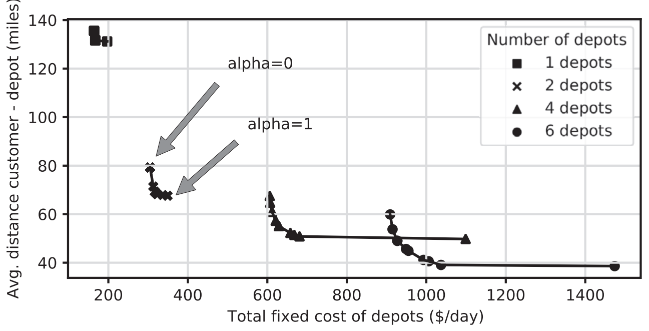

FIGURE 1 Visualization of the trade-off between optimizing for minimum cost of locating depots and minimum average distance customer-depot. Optimal solutions are obtained for combinations of 11 different, evenly spaced, values of $\alpha \in [0, 1]$ and four representative numbers of depots

not to customer nodes. The main difference with the p-median problem lies in constraints (28), which are needed to ensure that only nodes that can be *reached* from a depot can be assigned to that depot.

This p-median problem, with circa 1500 nodes or ZIP codes, is too large to solve to optimality with exact methods. Our approach consists of solving the p-median problem in two stages, considering different resolutions:

- *Stage 1*: Group nodes that are close together into clusters, forming at most 500 clusters. Then solve the p-median problem using these clusters instead of the original nodes to identify $|H|$ clusters that contain suitable depot locations. Since each of these $|H|$ clusters will generally include multiple nodes (multiple ZIP codes), Stage 2 assigns a depot to a single node in each of the clusters.
- *Stage 2*: Consider the p-median problem with the original nodes. Fix $z_j = 0$ for all nodes $j \in V$ that do not belong to the clusters identified in Stage 1. This ensures that only nodes that are in a cluster are candidate depot locations. Finally, solve the resulting p-median problem to obtain the final set of depot locations H.

Both steps run in less than a minute, while introducing only a small distortion to the spatial distribution of demand. To perform the clustering described in Stage 1, we apply k-medoids [16], an unsupervised machine learning technique similar to the well-known k-means algorithm [20]. For a given number of clusters k, k-medoids minimizes the dissimilarities between points in a cluster and the center of that cluster. It can be easily implemented if the input data is a distance matrix. The code to perform the clustering of nodes was written in Python using the library PyClustering [21].

Figure 1 illustrates the trade-off between depot costs and average distance, for a range of values of $\alpha \in [0, 1]$. It can be observed that as we prioritize minimizing the total fixed cost of depots ($\alpha \to 0$), the average distance depot-customer grows very rapidly. Similarly, if only the average distance metric is considered ($\alpha = 1$), then solutions with very high depot costs are obtained. However, by reducing α to 0.9, namely, by adding a small penalty for high depot costs, these costs are reduced between 20% and 40% while the increase in the average distance is negligible. This suggests that it is easy to find a cheaper alternative in the neighborhood of a location that is optimal under spatial considerations only.

Unlike routing costs, which are 100% opex, the fixed cost of the depots has a capex element (lease or purchase payments). This means that part of the depot costs will be amortized within a few years, whereas routing costs are running costs. Therefore, it seems sensible to invest in finding good locations for the depots at the expense of purchasing or leasing slightly more expensive facilities. For these reasons, we set α to 0.9. Using 2018 data, this scheme returned a depot configuration in 40 s on average, for different numbers of depots.

It should be noted that balancing of demand across different depots has not been included in this model. Having some variability in the volume of demand assigned to each depot is not considered to be problematic, since it is possible to assign fleets of different size to different depots.

6 | FLEET SIZING

The next strategic decision concerns the sizing of the fleet and the distribution of this fleet across different depots. This aspect of the problem is key to obtaining robust solutions. While a poor choice of depot locations or number of depots can be compensated with a larger fleet of vehicles, the fleet sizing is critical to ensuring all demand can be routed: too small a fleet will result in some customers not being visited on certain days unless additional resources are put in place.

To obtain a fleet size that can cope with 2018 demand but also be robust against unseen demand data (e.g., 2019 demand), it is not sufficient to set the fleet size to the number of vehicles required on the busiest day in 2018, because the fleet size required depends not only on the total demand but also on the spatial distribution of the customers. To overcome this, we propose to simulate a set of demand instances based on a busy period in 2018. We then determine the number of vehicles required on those simulated days by solving a VRP on each of these instances (using the methodology outlined in Section 8) and decide the fleet size based on these results.

Simulating "busy days" first requires an understanding of the demand distribution in 2018. Let $q(i, d)$ be the demand of customer i in day d. We assume that $q(i, d)$ is given by the product of two random variables $X(i, d)$ and $Y(i, d)$. $X(i, d)$ is a binary variable indicating whether customer i is visited on day d and $Y(i, d)$ indicates how many demand units customer i must be served on day d. The rationale behind this assumption is that we observed that, on average, only 5% of the customers had a positive demand ($q(i,j) > 0$) on any given day in 2018, and in those cases where $q(i,j) > 0$, this demand was variable and seemed to follow a distribution centered in 10.

Through statistical analysis we identified the following properties of X and Y that can be exploited to simulate instances that resemble the actual demand in 2018:

1. *The amount of goods required by an active customer node, $Y(i, d)$, is independent of the customer and the day.* $Y(i, d)$ can be replaced with Y.
2. *The amount of goods required by an active customer node, Y, follows a Poisson distribution with* $\lambda = 9.95$. The parameter λ was fitted using the maximum likelihood method. The goodness of fit was tested using a chi-squared test, and it can also be visualized in Figure 2A.
3. *$X(i, d)$ is a Bernouilli distributed random variable* that takes value 1 if customer i requires a visit on day d, which happens with probability p_{id}. This probability is different for every customer and day (see next point).
4. *The seasonality behind the demand in 2018 is driven by $X(i, d)$.* The R^2 score of a linear regression model linking the number of active customers with the total demand on a given day is 0.986, which indicates that 98.6% of the variability in the total demand is explained by the number of active customers. Busier days are so because there are more customers that have to be visited, and not because the demand per active customer is higher. The similarity between these two time series can be seen in Figure 2B.
5. With the exception of a few days in January, the *busiest period of the year is the summer season* (see Figure 2B). Also, a logistic regression model showed that *customers are 20.3% less likely to have demand on a weekend*.
6. *On any given day, the demands of different customers in terms of $X(i, d)$ are independent from each other.* The Pearson correlation coefficients ρ that resulted from comparing the demands of all pairs of customers are very low ($|\rho| < 0.15$ for 95% of all pairs of customers and max $|\rho| = 0.26$). As a result, we can expect that the spatial distribution of demand will be similar on different days. In other words, the relative volume of customers in each depot area will not vary significantly on different days. This is a convenient property from a fleet sizing point of view, given that vehicles must operate from the same depot every day.

Based on these observations, the following approach is used to simulate a "busy day":

1. *Sample from X*: each customer i is activated with a probability $p_{i,busyday}$. This probability is initially estimated as the frequency with which customer i required a visit over all weekdays from June to August in 2018. In other words, we sample data from an artificial "busy day" that behaves like an average summer weekday in terms of demand volume, and the probability $p_{i,busyday}$ will only depend on the customer node i. Furthermore, to achieve more robust solutions, customers that were not active during this period (and therefore would have zero chance of appearing in our simulated data) are assigned a probability p equal to the lowest nonzero probability in the data.
2. *Sample from Y*: for each customer that has been activated in the previous step, the amount of demand is sampled from *Poisson*(9.95).
3. *Calibrate demand level*: to obtain robust solutions that can ensure not only that all demand is satisfied in 2018, but also cope with a possible increase in demand in the future, we introduce a calibration factor $\beta > 1$ that increases the generated demand by multiplying $p_{i,busyday}$ by this factor at the time of simulating data. We choose to increase $p_{i,busyday}$ and not the parameter λ because it is variable X, and not Y, that is affected by seasonality.

In summary, the demand of customer i on a simulated "busy day" d is obtained from Equations (30)–(32):

$$X \sim \text{Bernouilli}(p_{i,busyday} \times \beta) \tag{30}$$

$$Y \sim \text{Poisson}(9.95) \tag{31}$$

$$q(i, d) = X \times Y \tag{32}$$

The value of β plays a crucial role in ensuring the robustness of our solutions; it will determine by how much we underestimate or overestimate the fleet size required. We tune this parameter by inspection to ensure that the average demand of the simulated instances is approximately equal to some demand level that we define in advance. In particular, we have explored

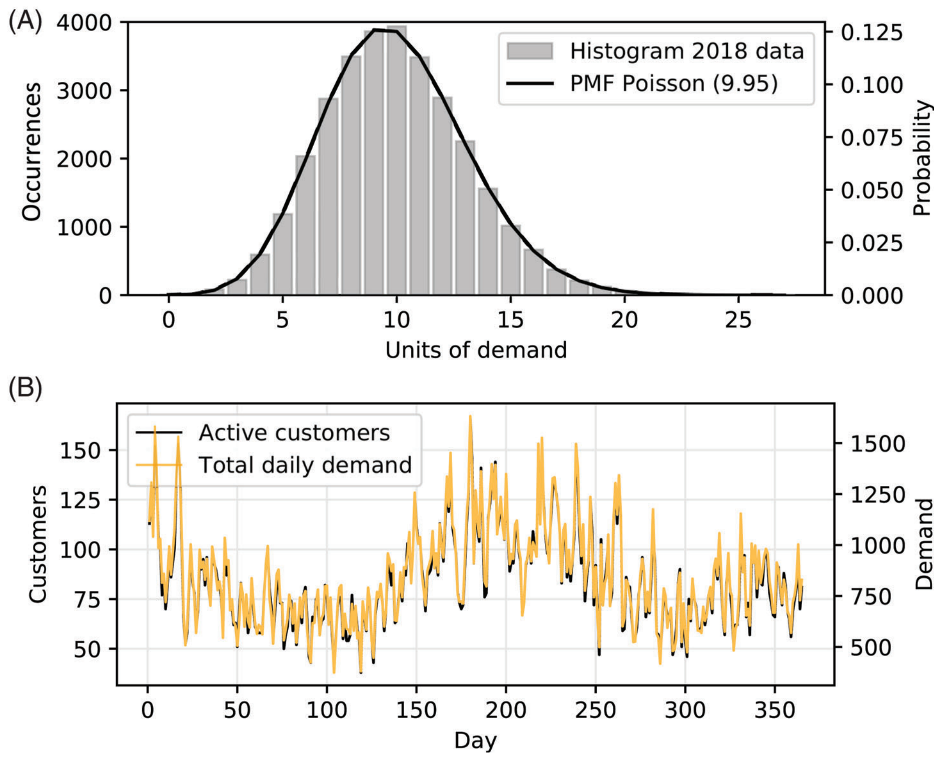

FIGURE 2 (A) Comparison between actual distribution of Y (units of demand per active customer) in 2018 and the Poisson distribution fitted to this data. (B) Total demand and number of active customers per day in 2018

TABLE 1 Estimated values of β to generate a set of instances with an average demand (in number of customers) approximately equal to four different levels of demand from 2018 data

Demand level	Active customers	Calibrated β
Max demand 2018 + 10%	184	1.7
Max demand 2018	167	1.55
P95 demand 2018	133	1.25
P75 demand 2018	115	1.05

several demand levels that may be of interest to the business: the maximum in 2018, the 95th percentile (P95) and 75th percentile (P75).

Finally, we solve the fleet sizing problem by completing the following steps:

Step 1. Choose a design criterion with its associated β from Table 1.
Step 2. Simulate n demand realizations with (30)–(32).
Step 3. For each generated instance, solve a VRP and store the number of vehicles required at each depot.
Step 4. Set the final fleet size assigned to each depot to the 95th percentile of the list of n vehicle requirements obtained in Step 3. This is different from the P95 used for simulating demand.

We choose to set the fleet size assigned to each depot to the 95th percentile of the list of n vehicle requirements to be conservative: it is sufficient to route the demand of 95% of the generated instances and it is less sensitive to extreme values in the results than the maximum requirement. We set the number of simulations n to 50, which is high enough to obtain a stable number of vehicles with the 95th percentile, but not too high that it takes a prohibitive amount of time to run the fleet sizing model. Lastly, we solve these 50 routing instances using our heuristic algorithm described in Section 8 with the standard savings function and no route improvements, all of which speed up the process and lead to more pessimistic and conservative results.

Results of running the fleet sizing algorithm for the four proposed demand levels in Table 1 are displayed in Figure 3. To simplify the analysis, we ran the algorithm for a single representative number of depots (four depots). The horizontal lines represent the fleet size obtained using different demand levels, whereas the time series represent the actual vehicle requirements on any day in 2018 and 2019.

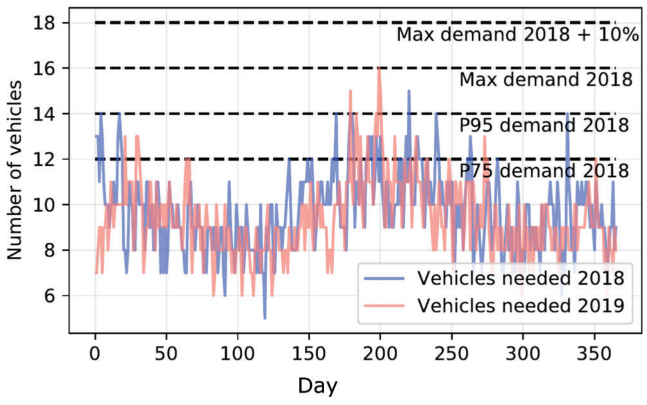

FIGURE 3 Minimum number of vehicles required to serve the demand everyday in 2018 and 2019, compared to the fleet size calculated using different design criteria. All the VRP instances considered in this analysis have been solved using the heuristic method from Section 8 to speed up the solution

TABLE 2 Comparison between different design criteria in terms of fleet size, fleet fixed costs, additional vehicles required when the existing fleet is insufficient to meet demand, associated costs of these extra vehicles and total vehicle costs, for each year

Design criterion	P75	P95	Max	Max + 10%
2018				
Vehicles assigned	12	14	16	18
Fixed vehicle costs	\$131k	\$153k	\$175k	\$197k
Extra vehicles needed	32	1	0	0
Cost of extra vehicles	\$2.9k	\$90	\$0	\$0
Total vehicle costs	\$134k	\$153k	\$175k	\$197k
2019				
Vehicles assigned	12	14	16	18
Fixed vehicle costs	\$131k	\$153k	\$175k	\$197k
Extra vehicles needed	26	4	0	0
Cost of extra vehicles	\$2.3k	\$360	\$0	\$0
Total vehicle costs	\$134k	\$154k	\$175k	\$197k

We can see that the same seasonality pattern is present in 2019 and there does not seem to be a positive trend in the year-over-year growth, although the maximum fleet requirement in 2019 (16 vehicles) was one unit higher than the highest requirement in 2018 (15 vehicles).

When the simulated demand is calibrated using the maximum demand in 2018, the resulting fleet size is sufficient to route all demand in 2018 and 2019, as can be seen in Figure 3. However, if the decision maker is risk-averse and does not have any contingency measures for when the existing fleet cannot cope with the upcoming demand, they should use the "maximum demand 2018 + 10%" criterion, which, in this case, gives two additional vehicles for safeguarding.

On the other hand, the P75 and P95 design criteria yielded fleet sizes that did not meet the vehicle demand in 2018 nor 2019. However, it is worth noting that the maximum demand in 2018 was significantly higher than average levels throughout the year: the maximum number of customers was 25% higher than the 95th percentile and 45% higher than the 75th percentile. This shows that designing for the peak will result in a low utilization of vehicles for most of the year. Therefore, if the business had the ability to rent additional vehicles for a day on short notice, there would be scope for potential savings by designing the fleet using the 75th percentile criterion. For the case presented in Figure 3, this resulted in a fleet of 12 vehicles, which suffices to serve the demand of most days.

To estimate the impact on costs of these different approaches, we present in Table 2 an analysis of the additional rental vehicles that would be required in 2018 and 2019 using different calibration criteria. To be conservative, we have assumed that the daily rate to rent a vehicle on short notice is triple that of standard vehicles, namely, \$90/day.

It is clear from these results that designing the fleet for peak demand results in a low utilization of vehicles for most of the year. If it is possible to rent vehicles on an ad hoc basis, even under our conservative assumption on rental costs, the cost of doing so using the P75 and P95 demand 2018 criteria would be negligible compared to the fixed vehicle costs.

Our approach to robust fleet sizing has the additional advantage that it allows the decision maker to manage risk and costs effectively. These criteria are intuitive and do not involve tuning a mysterious numeric parameter that is hard to interpret; β gets updated in the background. It took between 25 ("P75 demand") and 90 s ("Max demand + 10%") to run this fleet sizing model in a four-depot scenario.

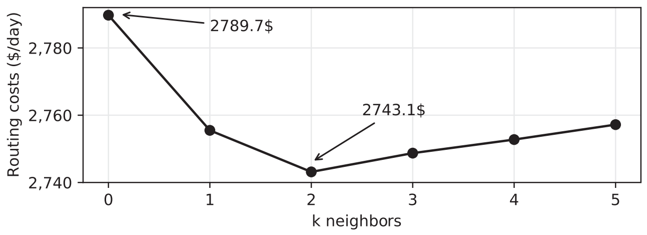

FIGURE 4 Impact on routing costs of including the average distance to the *k* nearest neighbors when assigning customers to depots. The results for the different values of *k* represent the average routing costs from 50 randomly selected instances with different numbers of depots

7 | ASSIGNMENT OF CUSTOMERS TO DEPOTS

We now enter the operational planning stage of the problem, which aims to determine the routes to be completed by each vehicle in each trip such that all the demand of a given day is served.

The mathematical model presented in Section 2 would simultaneously assign customers to depots and determine the optimal routes. However, this would require to solve a large multi-depot VRP. To overcome this, we assign customers to depots a priori. This allows us to treat the routing problem as a set of independent, smaller VRP's, one for each depot.

A reasonable approach to achieve this would be to assign customers to their closest depot for any demand realization. However, this does not take into account the spatial distribution of the demand on a given day, for example, it might be "easier" to visit a customer by a vehicle completing a trip nearby, even if this vehicle is based on a depot that is not the closest to the customer.

For this reason, in this section we explore the effect of factoring in the "routing effort" required to visit a customer. Jarrah and Bard [14] designed a metric based on the intra-customer travel time to address the problem of geographic clustering of customers. They estimate the travel time from customer *i* to the next customer based on the distance to the *k* nearest customers in the same cluster.

We have explored this idea of the distance to the *k* nearest customers as a proxy of the routing effort in the problem of assigning customers to depots on a given day. The approach we propose is as follows: starting from an initial assignment of customers to their closest depots, we iterate over the set of customers and we assign each customer $i \in V$ to a depot $\ell \in H$ such that the average distance from *i* to the *k* nearest neighbors currently assigned to depot ℓ plus the distance from *i* to the depot is minimum. This metric is

$$\text{depot of customer } i \equiv \ell^* = \arg\min_{\ell} \left\{ d_{\ell i} + \frac{1}{k} \sum_{j \in V_\ell^k} d_{ij} \right\}, \tag{33}$$

where V_ℓ^k is the set *k* customers currently assigned to depot ℓ that are the closest to customer *i*. Because every time we change the depot assigned to a customer this might change the *k* nearest neighbors of customers that we have already scanned, we repeat this process iteratively until the assignments are stable, that is, until there is no change in the assignment of any customer in one round.

Including the distance to the depot in Equation (33) is necessary to ensure that, in a stable solution, the clusters of customers are still centered around the depots. Without this term, the iterative assignment process could converge in $|H|$ clusters of customers with no connection with the depot locations. This was not considered in [14], since this study was only concerned with cluster construction.

It only remains to choose a value of *k*. Figure 4 shows routing costs on randomly selected instances and the impact of parameter *k*. Including the nearest neighbors in the assignment always results in savings in routing costs for the values of *k* tested in this experiment. With $k = 2$, there is a cost reduction of circa $45/day with respect to assigning customers to the closest depot ($k = 0$). Although this value is small compared to the daily routing costs, it adds up to an annual $18k saved on routing costs and it can be achieved with very little computational effort.

8 | VEHICLE ROUTING

At this point, the original routing problem has been divided into $|H|$ subproblems (one for each depot) of a more manageable size. These subproblems belong to the class of *Multi-Trip Vehicle Routing Problems with Time Windows* (MTVRPTW) [2, 5]. There is now a single depot in each subproblem and a known fleet size denoted by $|K|$. The mathematical model in Section 2 can be employed to solve some of these instances, with some considerations:

1. Decision variables z_i, which determine the location of depots, are now known and can be treated as problem data ($z_i = 1$ if customer i is the depot of a given subproblem, 0 otherwise).
2. Constraint (7), which fixes the number of depots to $|H|$, is no longer needed.
3. Constraints (12), which ensure that the same vehicle cannot complete two trips in two different depots, can also be removed.
4. We set the cardinality of the set of vehicles K to the number of vehicles assigned to the depot to ensure the number of vehicles used does not exceed the fleet size. Even though the daily fleet costs are fixed regardless of the daily vehicle utilization, we keep them in the objective function to encourage solutions with high fleet utilization. This avoids solutions where many or all the available vehicles are assigned to one or two short trips, which would not be desirable from a driver rostering perspective. Given the low cost of the vehicles, it is safe to add fleet costs in the objective function without significantly affecting the routing costs, which are the main cost component to minimize at this stage.

The computational experiments carried out to test the limits of this mathematical model, presented in Section 8.3 below, revealed the need for a more sophisticated exact method for smaller instances and a heuristic approach that can handle the larger instances in reasonable time at the expense of generating suboptimal routings in some cases. Based on these results, we adopt a hybrid approach: on any day, instances (or depots) with fewer than 15 customers are solved using the branch-and-cut algorithm described in Section 8.1 with a time limit of 20 min; for larger instances we resort to the heuristic method described in Section 8.2. This time limit of 20 min is high enough to obtain acceptable solutions for small instances but low enough that routes can be generated in an acceptable amount of time after the demand data becomes available at 6:00 p.m.

8.1 | Exact method

In an effort to improve the efficiency of the exact method implemented by CPLEX, we propose a VRP-specific branch-and-cut approach. For this purpose, we define a new set of valid inequalities that are added sequentially in the form of cuts in the branching tree. We use the dual simplex method to solve the linear programming (LP) relaxation of the original MIP problem at every node, disable all of the built-in cuts used by CPLEX, apply a coefficient reduction technique and use our heuristic solution to warm start the MIP.

We ran a series of experiments to decide on the most appropriate method to apply the cuts throughout our branching tree, while balancing the time needed for separation, the number of additional constraints and the rate of improvement. As a result of our experiments, we perform a separation procedure in 10 consecutive nodes in the branching tree, remove all possible violations, and repeat this separation once in every 500 inspected nodes. This way, when the algorithm enters the 1500th node, only 30 nodes will have been inspected for violations, and the risk of adding an overwhelming number of cuts that may slow down the solution process is lowered. All the cuts added are globally valid, and the results obtained are presented in Section 8.3.

8.1.1 | Valid inequalities

If the depot nodes are excluded from the graph of our problem, a feasible solution should only consist of *open paths*. Therefore, any cycle in this graph represents a violation of a constraint of the problem and should be eliminated. This logic leads to the most common valid inequalities for the TSP, known as subtour elimination constraints. Even though in our case the time window constraints (15)–(21) prevent subtours in any integer solution, the fractional solutions resulting from solving the LP relaxation of the problem can easily contain subtours. By implementing an adequate separation procedure, one is able to identify these violations and apply an appropriate constraint that is globally valid, thus making this solution infeasible in all future nodes. The constraints known as *lifted subtour inequalities* [12] are facet defining for the TSP for $|S| \geq 3$, $S \subseteq V$. These inequalities have shown good results when applied to the VRPTW [4] and will be used to tighten the polyhedron of the LP relaxation of our problem. Due to the form of our degree constraints (2) and (3) and the fact that we are allowing multiple trips per vehicle, we need to implement the lifted subtour inequalities in a different form:

$$\sum_{\substack{k\in K\\ r\in R}} \sum_{j=1}^{|S|-1} x_{i_j i_{j+1}}^{kr} + \sum_{\substack{k\in K\\ r\in R}} x_{i_{|S|} i_1}^{kr} + 2\sum_{\substack{k\in K\\ r\in R}} \sum_{j=2}^{|S|-1} x_{i_j i_1}^{kr} + \sum_{\substack{k\in K\\ r\in R}} \sum_{j=3}^{|S|-1} \sum_{h=2}^{j-1} x_{i_j i_h}^{kr} \leq |S| - 1, \qquad S \subseteq V, |S| \geq 3 \tag{34}$$

$$\sum_{\substack{k\in K\\ r\in R}} \sum_{j=1}^{|S|-1} x_{i_j i_{j+1}}^{kr} + \sum_{\substack{k\in K\\ r\in R}} x_{i_{|S|} i_1}^{kr} + 2\sum_{\substack{k\in K\\ r\in R}} \sum_{j=3}^{|S|} x_{i_j i_1}^{kr} + \sum_{\substack{k\in K\\ r\in R}} \sum_{j=4}^{|S|} \sum_{h=3}^{j-1} x_{i_j i_h}^{kr} \leq |S| - 1, \qquad S \subseteq V, |S| \geq 3 \tag{35}$$

The subtours of size $S = \{i, j\}$ (|S|=2) will be cut off using

$$\sum_{\substack{k\in K\\ r\in R}} x_{ij}^{kr} + \sum_{\substack{k\in K\\ r\in R}} x_{ji}^{kr} \leq 1 \tag{36}$$

8.1.2 | Separation procedure

Once a feasible solution for the LP relaxation has been found, the edges from the set E in the graph $G(V, E)$, where V is the set of original customers, are assigned a weight $w(i,j) = \sum_{k\in K}\sum_{r\in R} x_{ij}^{kr}$. To separate the violated subtour inequalities (34)–(36), we use a procedure called *shrinking*. A detailed description of this procedure applied to the TSP can be found in Lawler [19], and similar procedures are commonly used to separate this kind of inequalities. The separation procedure consists of creating additional nodes in the graph, while simultaneously removing both end nodes of an edge with positive weight. Each new node created in the graph is a *child* node, and the nodes that are removed during this process are its *predecessors*. That way, each child node has a line of predecessors that can be backtracked to the set of original nodes that generated it. Let $n = |V|$ and assume a feasible solution has been obtained for the LP relaxation. The procedure is as follows:

- Step 0. For each node $i \in V$ initiate an empty set of predecessors.
- Step 1. Find an edge (i,j) such that $w(i,j) > 0.5$ and $w(j,i) > 0.5$. If such edge exists, go to Step 5. Otherwise, go to Step 2.
- Step 2. Find an edge (i,j) such that $w(i,j) > 0$. If such edge exists, go to Step 3. Else, **Stop**.
- Step 3. Create child node $\ell := n + 1$, and update the weights $w(s, \ell) := w(s, i) + w(s, j)$ and $w(\ell, s) := w(i, s) + w(j, s)$, for every $s \in V, s \neq i, j$. Update $n := n + 1$.
- Step 4. Remove ℓ's predecessors i and j from V ($V := V \setminus \{i, j\}$). Update the set of ℓ's predecessors with nodes i and j and all of their own predecessors. Go to Step 1.
- Step 5. Create a set S of original nodes containing i and j with their predecessors. If $|S| = 2$, insert cut (36). If $|S| > 2$, insert both cuts (34) and (35). Reoptimize.

This procedure is illustrated in Figure 5. From part 5(F), we can see that the set S of original nodes contains nodes $i = 6$ and $j = 4$ with their predecessors: $S = \{1, 2, 3, 4\}$. Note that node $i = 6$ is not in S as it is not an original node.

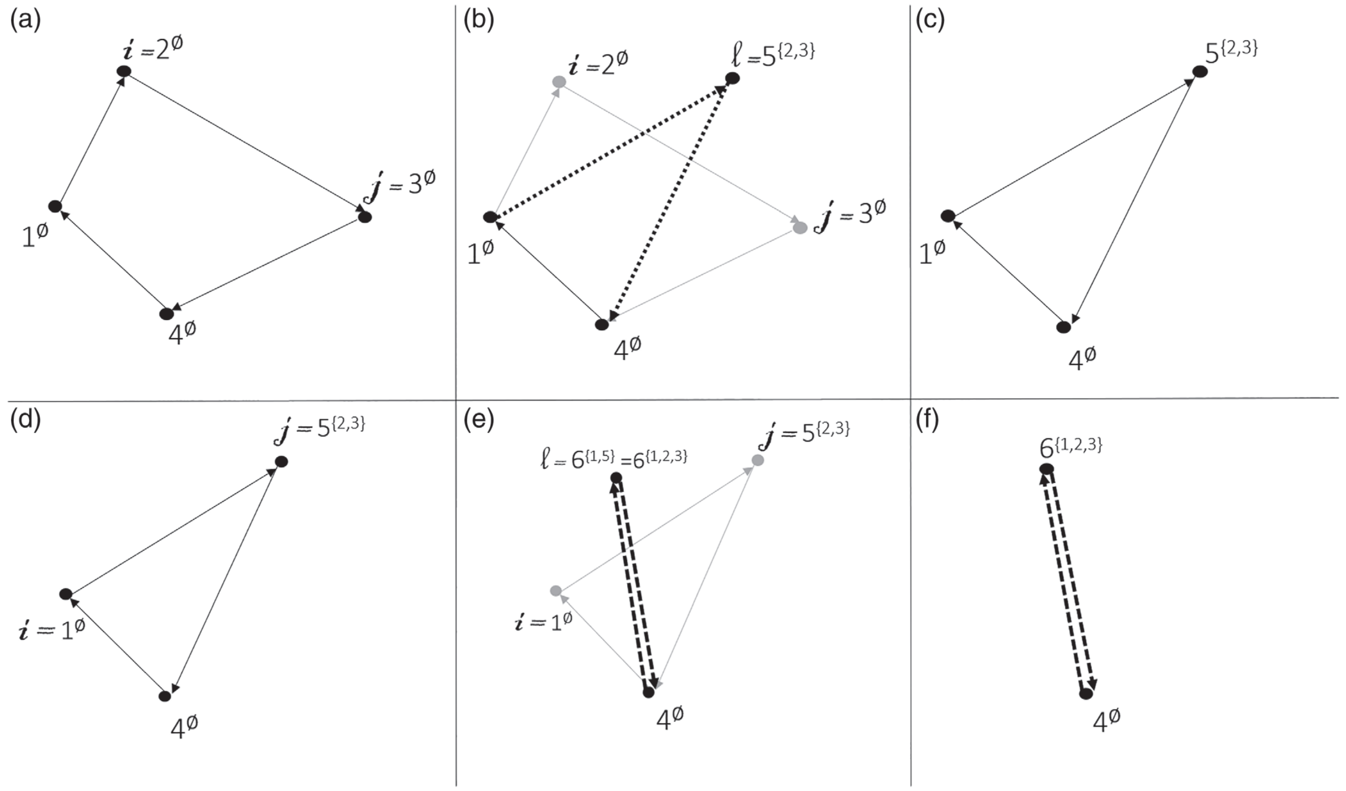

FIGURE 5 Graphical representation of the separation procedure: (A) Step 0, (B) Step 1 → Step 2 → Step 3, (C) Step 4, (D) Step 1 → Step 2 → Step 3, (E) Step 4, (F) Step 1 → Step 5

8.2 | Routing heuristic

For instances with more than 15 customer nodes, we have developed a heuristic algorithm. This heuristic has three stages: first, the multi-trip characteristic is ignored and routes are constructed considering only vehicle capacities and time windows. Next, solution improvement techniques are employed to improve the routes constructed in the previous step. Lastly, a packing algorithm assigns routes to trips of vehicles.

8.2.1 | Route construction heuristic

Our route construction approach is inspired by the adaptive routing heuristic proposed by Batarra et al. in [5]. The authors first determine the period of time during which each route is strongly active. A route is strongly active at some time interval if it is guaranteed that the route will be in progress at that time, regardless of when it starts see Figure 6.

Since the number of routes that are strongly active at the same time define a lower bound on the number of vehicles required, Batarra et al. [5] penalize routes that are strongly active at the busiest time, the *critical time*, and adjust the insertion costs in their route construction heuristic to include these penalties. The expected result is that routes constructed using these penalties will avoid the critical times and will therefore be easier to pack in the packing stage of the algorithm. They repeat this two-stage process iteratively (constructing routes – calculating penalties) until some stopping condition is met.

In the VRPTW considered in this work, all customers can be visited between 8:00 a.m. and 4:00 p.m. and therefore the critical time will always occur around the center of this time window. This is not the case in [5], where different customers can have different time windows and therefore critical times may occur at any time of the day. We take advantage of this property of our problem to introduce two substantial simplifications to the approach in [5]: first, the penalty for the critical time interval is a fixed value that we set in advance; second, routes are only constructed once, including the penalties.

We propose a parametric savings heuristic based on the general parallel savings heuristic developed by Clarke and Wright [7]. We define the savings function as follows:

$$s(i,j) = s_{routingtime}(i,j) + \lambda \; s_{stronglyactive}(i,j) \tag{37}$$

The first term is the well-known savings function, namely, $s_{routingtime}(i,j) = t_{0i} + t_{j0} - t_{ij}$. It represents the savings in traveling time when routing a vehicle from depot 0 to customer i to customer j and back to the depot as opposed to visiting i and j alone in different trips. We introduce a second term, $s_{stronglyactive}(i,j)$, which represents the benefit of joining route finishing in customer i, with route starting with customer j in terms of how "easy" the resulting route would be to pack compared to how "easy" it would be to pack the individual routes. $s_{stronglyactive}(i,j)$ is defined in Equation (38):

$$s_{stronglyactive}(i,j) = f_{correctedSA}(\text{route with i and j}) - f_{correctedSA}(\text{route of } \; i) - f_{correctedSA}(\text{route of } \; j). \tag{38}$$

The term $f_{correctedSA}$ represents a function that penalizes routes that are strongly active at some time. We introduce here an additional modification with respect to the approach in [5]. We seek to penalize the amount of strongly active time only up to a certain point. As the duration of a route starts to approach the vehicle time window length, that is, 11 h, it becomes more convenient to keep expanding this route, because the remaining free time of the vehicle would become more and more difficult to fill by another route. In other words, it is the routes of medium duration which can make the packing of routes in vehicles inefficient. Using this logic, we define $f_{correctedSA}$ as follows:

$$f_{correctedSA} = \min\{11\text{ h} - f_{SA},\ f_{SA}\} \tag{39}$$

where f_{SA} is simply the amount of time that a route is strongly active. Based on this definition, $f_{correctedSA} = 0$ when a route is not strongly active at any time ($f_{SA} = 0$), it increases linearly reaching a maximum when $f_{SA} = 5.5$ h, at which point it decreases

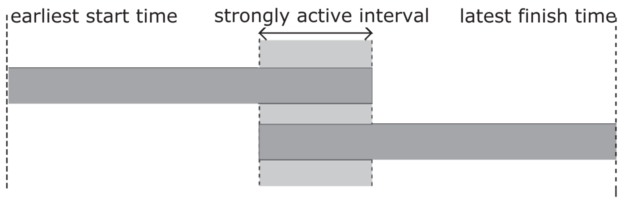

FIGURE 6 Graphical definition of the strongly active interval of a route

until $f_{SA} = 11$ h, where $f_{correctedSA}$ is equal to 0 again. This means that no penalty is applied to routes that are not strongly active at any time nor routes that are 11 h long.

Lastly, the term λ in (37) controls the weight of the term $s_{stronglyactive}(i,j)$ in the savings function and it has been set to 0.05 through computational experiments.

The rest of the algorithm follows the same structure as the parallel savings algorithm by Clarke and Wright [7]. Pairs of customers (links) are sorted in decreasing order by their savings. Starting from the first link on the list, links are used to join smaller routes into larger routes as long as they are feasible in terms of vehicle capacity and time window restrictions. The algorithm stops when the list contains no more feasible links.

Including the term $s_{stronglyactive}$ to the savings function does add a small amount of computational time, as this term varies with the duration of the routes, and the savings list needs to be updated multiple times. Nevertheless, the total time in the worst case scenario, a single depot on a "busy day," was under 6 s.

8.2.2 | Solution improvement

After constructing the initial routes, two solution improvement techniques are employed to possibly improve the current routes by reducing the routing costs. First, a TSP with time windows (TSPTW) [11] is solved to optimality for each of the constructed routes to ensure customers are visited in the shortest possible path within the route. For the sake of brevity, the MILP formulation of this TSPTW is not presented here. Last, we apply an exchange heuristic where we swap two customers visited in different routes with the aim to reduce the routing distances further [23].

8.2.3 | Packing heuristic

The heuristic we propose is a modification of the well-known *first fit decreasing* (FFD) heuristic [10], which sorts items in descending order of size, takes one item at a time and either assigns it to the first bin where it fits or opens a new bin if it did not fit in any of the open bins.

We propose a variation of the FFD heuristic in which the items (the routes) are also sorted in decreasing order by their size (route duration) and they are assigned to the first vehicle in which they fit, at the earliest possible start time, such that the time windows are not violated. The main difference with the FFD is that, when we assign a route to one trip to a vehicle, we pay attention to *where* in the current schedule of that vehicle it gets inserted.

We will insert the route at the start of the schedule if doing so reduces the difference between the depot opening time and the earliest start time of the first trip currently scheduled on that vehicle. If that is not the case, we insert it at the end if the route has a longer travel time from the last customer to the depot than the current last trip. Otherwise, we insert it in any intermediate position. The rationale for this modification of the FFD algorithm is that we want to maximize the use of the 2-h and 1-h gaps at the start and end of the day, respectively, where the depot is open but customers cannot be served.

Furthermore, we observe that most routes can be operated in either direction, namely, customers can be visited in reversed order. This is true because the time windows of all customers are identical and the graph is undirected. The only case in which a route cannot be reversed is illustrated in Figure 7B. In light of this observation, we allow routes to be inserted in reverse order if it is feasible and beneficial.

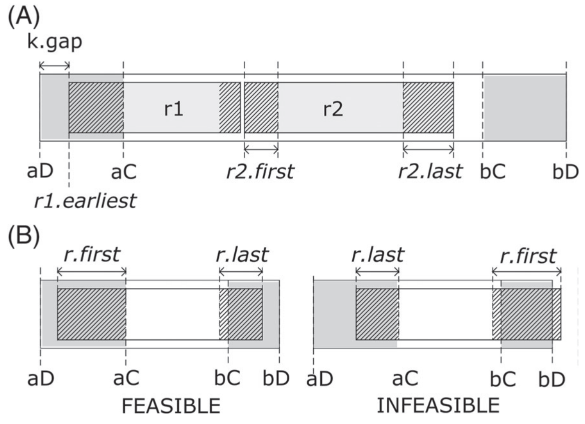

FIGURE 7 (A) Illustrative schedule with two trips displaying problem parameters. (B) Example of a case where inverting the direction of the trip results in an infeasible schedule because the vehicle returns to the depot after b_D

A detailed description of this heuristic is presented in Algorithm 1 in the Appendix. Lastly, we define the key parameters of this packing heuristic below, which are used in Algorithm 1. These definitions are accompanied by a graphical description in Figure 7A.

Parameters:

a_D	Opening time of depot
b_D	Closing time of depot
a_C	Earliest customer visit allowed
b_C	Latest customer visit allowed
r.first	Travel time between depot and first customer of trip *r*
r.last	Travel time between last customer and depot of trip *r*
r.earliest	Earliest time that trip *r* could be scheduled to start, namely, $\max\{a_D,\ a_C - r.first\}$
k.gap	Time gap between a_D and the earliest start time (*r.earliest*) of the first trip for vehicle *k*

8.3 | Comparison of routing methods

Table 3 shows the results obtained using (i) the branch-and-cut algorithm with an initial heuristic solution and our subtour elimination constraints as the only cuts (34)–(36), (ii) the CPLEX default settings, which automatically decide between dynamic search and branch-and-cut with default CPLEX MIP cuts, depending on the model, and (iii) our heuristic method. The number of cuts found by our separation procedure is also shown for every instance. In both exact methods, the optimality gap tolerance was set to 0.001. Other variants of our branch-and-cut algorithm were also explored, namely, a combination of subtour elimination cuts and CPLEX cuts (with and without an initial heuristic solution) and subtour elimination constraints without warm start. However, we have not included these results in Table 3, as these algorithms were outperformed by our final exact method.

Our heuristic method obtained optimal routing costs in all instances with less than 15 customers and it often outperformed both exact methods with a 20-min time limit for larger instances. The number of vehicles obtained is different in some cases, but there does not seem to be a method that consistently requires more vehicles.

The comparison between our branch-and-cut algorithm and CPLEX with default settings looks promising. Even though this comparison gives CPLEX an advantage, our branch-and-cut algorithm outperformed CPLEX in many instances and when CPLEX's results were better, this difference was relatively small. Furthermore, we ran several experiments comparing our

TABLE 3 Computational results of several real instances in January 2018 ranging from 6 to 22 customers

	Subtour elimination cuts with warm start					Default CPLEX settings				Heuristic		
Number of customers (#)	Routing cost ($)	MIP gap (%)	Cuts added (#)	Vehicles (#)	Time (s)	Routing cost ($)	MIP gap (%)	Vehicles (#)	Time (s)	Routing cost ($)	Vehicles (#)	Time (s)
6	**153.1**	≈0	6	1	0	**153.1**	≈0	1	0	153.1	1	<1
8	**148.0**	≈0	50	1	2	**148.0**	≈0	1	0	148.0	1	<1
9	**185.5**	≈0	10	1	7	**185.5**	≈0	1	1	185.5	1	<1
9	**154.0**	≈0	115	1	4	**154.0**	≈0	1	1	154.0	1	<1
10	**190.8**	≈0	46	1	8	**190.8**	≈0	1	2	190.8	1	<1
11	**136.2**	≈0	1894	1	266	**136.2**	≈0	1	240	136.2	1	<1
12	**216.8**	≈0	418	1	39	**216.8**	≈0	1	32	216.8	1	<1
15	**306.4**	13.4	2448	2	1200	**306.4**	11.9	2	1200	306.4	2	<1
15	**246.2**	22.6	25 919	1	1200	**246.2**	12.1	1	1200	253.6	2	<1
15	**309.1**	30.5	13 821	2	1200	**309.1**	28.0	2	1200	309.7	2	<1
17	**275.0**	34.6	12 499	2	1200	**275.0**	34.2	2	1200	311.6	2	<1
18	**386.5**	22.1	36 389	2	1200	391.4	21.2	2	1200	386.5	2	<1
18	303.8	32.5	13 618	2	1200	**298.4**	26.3	1	1200	300.7	2	<1
19	**244.0**	28.9	11 896	2	1200	246.6	19.8	2	1200	251.9	1	<1
19	**391.5**	75.5	25 862	1	1200	395.0	22.4	1	1200	391.5	2	<1
21	**148.1**	33.2	26 818	1	1200	**148.1**	30.2	1	1200	148.1	1	<1
21	**222.7**	25.6	21 431	1	1200	223.8	24.3	2	1200	232.0	1	<1
22	1511.1	34.4	16 477	2	1200	**1495.1**	75.8	2	1200	474.4	2	<1
22	**374.8**	35.3	8641	2	1200	414.5	27.1	2	1200	372.9	2	<1

Notes: Results compare our branch-and-cut algorithm with subtour elimination cuts and warm started with a heuristic solution, the default strategy in CPLEX and our routing heuristic. The time limit for the exact methods was set to 1200 s, as per our proposed routing method, and the optimality gap tolerance was set to 0.001. Results in bold are the lowest routing costs of the two exact methods.

TABLE 4 Run times of 50 routing instances (10 for each range of number of customers) in 2018 ranging from 25 to 150 customers solved using our routing heuristic

Number of customers	26–50	51–75	76–100	101–125	126–150
Avg. run time (s)	2	7	20	38	80
Max. run time (s)	3	9	24	51	111

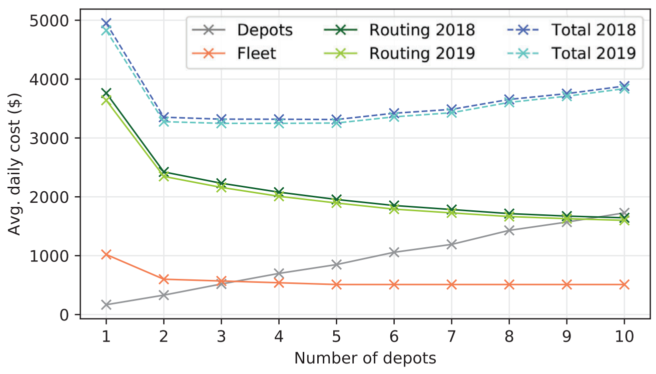

FIGURE 8 Average daily costs in 2018 and 2019 for 1–10 depots, broken down into depot, fleet and routing costs. Routing problems are solved for all days in the data sets provided using the heuristic method to speed up the solutions

branch-and-cut algorithm to a branch-and-bound with no cuts. We observed that ours performed significantly better, highlighting the efficiency of our adapted subtour elimination cuts.

Another observation is that our algorithm seems to reduce the peak memory used. Based on several experiments run in AIMMS for instances of up to 25 customers, we observed that after 20 min the peak memory was over 60% lower with our subtour elimination cuts with an initial heuristic solution compared to CPLEX's default.

Lastly, Table 4 presents a separate computational experiment run on larger instances with 25–150 customers (the highest number of customers in a single day), solved using the routing heuristic. The average run time is 30 s and all instances were solved in under 2 min.

9 | SOLUTIONS

Solutions have been generated for each day in the provided data sets and every number of depots from 1 to 10. The resulting costs are summarized in Figure 8. Due to the long computational times that would have been required to run our hybrid routing algorithm on 730 days, these routing problems have been solved using the heuristic method only. They represent a worst case scenario, as using an exact method for small instances can only improve these results.

There is a clear trade-off between depot costs and fleet and routing costs. As expected, fleet costs contribute the least to the total daily cost due to their low price, and they remain almost constant from five depots onward. Routing costs are the only cost component that varies on a daily basis and as such, average routing costs are different in 2018 and in 2019. The routing costs in 2019 are slightly lower, although the difference is not significant. Two conclusions can be drawn from this: 2019 demand did not change substantially with respect to 2018 and our methods produce solutions that are robust against unseen data.

Lastly, it is clear that a single depot is not cost-effective and the lowest total costs are obtained for two to five depots. This may be explained by the fact that there are two major cities in the State of Pennsylvania: Pittsburgh and Philadelphia, as illustrated in Figure A1a. These urban areas concentrate a significant fraction of the demand such that once a depot is placed in each of these city areas, the marginal benefit of additional depots is greatly reduced.

10 | OPTIMIZING THE NUMBER OF DEPOTS

In this section, we develop a framework for automating the choice of the number of depots making use of our existing models and performing an enumeration. To obtain a solution in a reasonable amount of time, we simplified our methodology by (i) halving the number of simulated days needed to determine the fleet size; (ii) designing the fleet for the "P75 demand in 2018"; and (iii) using for the routing stage the same fast routing approach employed in the fleet sizing model. Furthermore, instead of considering all days in 2018, we selected 15 at random to obtain an average daily routing cost.

Taking into account the trade-off between depot costs and fleet and routing costs, we designed the following iterative scheme. We compute the total costs (depot, fleet and routing) for different numbers of depots, starting from one and increasing the number of depots by one at a time. Every time we obtain a total cost for $|H|$ depots, we compare with the previous cost (for $|H| - 1$). If the new total cost is lower, we move on to $|H| + 1$ depots; otherwise, we stop: we can conclude that the optimal choice of depots is $|H|$. Using 2018 demand data, this scheme returned a configuration with five depots in just over 4 min of run time.

11 | CONCLUSIONS AND RECOMMENDATIONS

We have presented a mathematical model and a solution approach for the multi-trip VRP with time windows, fleet sizing and depot location entered into the 12th AIMMS-MOPTA Optimization Modeling Competition.

Given the complexity of the problem at hand, and assuming a business need to separate strategic and operational decisions, we decomposed this VRP into simpler problems that we solve sequentially. First, we address the location of depots, followed by the fleet sizing (strategic level); then, we determine the routings of a given day (operational level).

The depot location has been treated as a p-median problem. We narrowed down the search by clustering customer nodes using k-medoids as an intermediate step. To determine the fleet size at each depot, we developed a robust methodology that exploited the statistical properties of the demand data to generate feasible solutions for 2018 and 2019. In addition, we give the user the ability to adjust the demand coverage for the level of risk that they are willing to take.

The assignment of customers to depots is decided daily and it takes into account the distance to the depot and the "routing effort" of a customer node from different depots. It was shown that this yields lower routing costs than assigning customers to their nearest depot. For the vehicle routings, we proposed a hybrid approach, namely, we use an exact method for small instances and a fast heuristic for instances with more than 15 customers.

The exact method uses CPLEX and a VRP-specific branch-and-cut algorithm using lifted subtour inequalities, aiming to increase the number of instances on which the exact method can be applied. For the vehicle routing heuristic, we proposed a parametric savings heuristic to encourage routes that are easy to "pack" into vehicles and we developed a packing algorithm that exploits the structure defined by the time windows.

For any number of depots, we were able to determine depot locations, fleet sizes, and vehicle routings to serve the demand of each day in 2018 and 2019. We compared several routing methods and the results are promising. Our heuristic method often obtains optimal routing costs in small instances, suggesting that it may provide acceptable results in larger ones. The VRP-specific cuts used outperformed CPLEX in some instances, providing acceptable results in all other instances. These results validate our hybrid approach to the routing problem, and suggest that it is worth exploring the introduction of other VRP-specific cuts to further enhance our method.

Lastly, based on the analysis presented in this article, we make the following recommendations:

1. *There should be a minimum of two depots*. A single depot would result in a 47% increase in daily costs with respect to two depots due to increased traveling distances.
2. *We recommend to place five depots*. The total costs are very similar for any number of depots between two and five; however, five depots are preferred as this results in the lowest routing costs. This is interesting for two reasons: (i) it leads to lower carbon emissions and (ii) routing costs are 100% opex, whereas depots have a capex element (e.g., lease or purchase of depots) that can be amortized over time.
3. To reduce fleet costs, it would be worthwhile to *design the fleet size assigned to each depot based on the 75th percentile demand in 2018 and to rent additional vehicles on peak days*. If this is not a viable option, the fleet should be designed using the "max. demand in 2018 + 10%" criterion instead, to ensure complete coverage of demand.
4. The current assumption that no time is required for loading and unloading at the depot nor at a customer location may lead to unplanned driver overtime and customers not receiving their deliveries by 4:00 p.m. As an example, in a four-depot scenario with 8.7 customers visited per vehicle in a day and assuming 15 min per customer, this would result in two additional hours of driver time per vehicle per day. Two visits to the depot could add 1 h on top of that. *We therefore recommend that reasonable values for these parameters be included in the routing model.*

ACKNOWLEDGMENTS

The authors thank Boris Defourny, from Lehigh University, and the AIMMS team for the support provided during the AIMMS-MOPTA competition. We are also thankful to Sergio García Quiles and Jörg Kalcsics, from the University of Edinburgh, for their useful feedback, and last but not least, to the anonymous reviewer, whose insightful comments helped us to considerably improve the quality of this article. Albert Solà Vilalta was supported by The Maxwell Institute Graduate School in Analysis and its Applications, a Centre for Doctoral Training funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016508/01), the Scottish Funding Council, Heriot-Watt University, and the University of Edinburgh.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the AIMMS-MOPTA competition organizers. Restrictions apply to the availability of these data, which were used under license for this study. Data are available at https://coral.ise.lehigh.edu/~mopta/competition# with the permission of the AIMMS-MOPTA competition organizers.

ORCID

Paula Fermín Cueto https://orcid.org/0000-0003-0699-1207
Ivona Gjeroska https://orcid.org/0000-0001-6761-4005
Albert Solà Vilalta https://orcid.org/0000-0002-1248-5784
Miguel F. Anjos https://orcid.org/0000-0002-8258-9116

REFERENCES

[1] AIMMS-MOPTA Optimization Modeling Competition, 2020, available at https://coral.ise.lehigh.edu/~mopta/competition (Accessed November 18, 2020).
[2] N. Azi, M. Gendreau, and J. Y. Potvin, *An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles*, Eur. J. Oper. Res. **202** (2010), 756–763.
[3] R. Baldacci, M. Batarra, and D. Vigo, *Valid inequalities for the fleet size and mix vehicle routing problem with fixed costs*, Networks **54** (2009), 178–189.
[4] J. F. Bard, G. Kontoravdis, and G. Yu, *A branch-and-cut procedure for the vehicle routing problem with time windows*, Transp. Sci. **36** (2002), 250–269.
[5] M. Battarra, M. Monaci, and D. Vigo, *An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem*, Comput. Oper. Res. **36** (2009), 3041–3050.
[6] J. Brandão and A. Mercer, *The multi trip vehicle routing problem*, J. Oper. Res. Soc. **49** (1998), 799–805.
[7] G. Clarke and J. W. Wright, *Scheduling of vehicles from a central depot to a number of delivery points*, Oper. Res. **12** (1964), 568–581.
[8] M. S. Daskin and K. L. Maass, *The p-median problem*, in *Location Science*, G. Laporte, S. Nickel, and F. Saldanha da Gama, Eds., Springer International Publishing, Basel, 2015, 21–45.
[9] E. W. Dijkstra, *A note on two problems in connexion with graphs*, Numer. Math. **1** (1959), 269–271.
[10] G. Dósa, *The tight bound of first fit decreasing bin-packing algorithm is FFD(I) ≤ 11/9 OPT(I) + 6/9*, in *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, B. Chen, M. Paterson, and G. Zhang, Eds., Springer-Verlag, Berlin Heidelberg, 2007, 1–11.
[11] Y. Dumas, J. Desrosiers, E. Gelinas, and M. M. Solomon, *An optimal algorithm for the traveling salesman problem with time windows*, Oper. Res. **43** (1995), 367–371.
[12] M. Fischetti and P. Toth, *A polyhedral approach to the asymmetric traveling salesman problem*, Manag. Sci. **43** (1997), 1520–1536.
[13] B. Golden, A. Assad, L. Levy, and F. Gheysens, *The fleet size and mix vehicle routing problem*, Comput. Oper. Res. **11** (1984), 46–66.
[14] A. I. Jarrah and J. F. Bard, *Pickup and delivery network segmentation using contiguous geographic clustering*, J. Oper. Res. Soc. **62** (2011), 1827–1843.
[15] B. Kallehauge, J. Larsen, O. B. Madsen, and M. M. Solomon, *Vehicle routing problem with time windows*, in *Column Generation*, G. Desaulniers, J. Desrosiers, and M. M. Solomon, Eds., Springer US, Boston, 2005, 67–98.
[16] L. Kaufman and P. Rousseeuw, *Finding groups in data: An introduction to cluster analysis*, John Wiley & Sons, Hoboken, 1990.
[17] G. Laporte, *Fifty years of vehicle routing*, Transp. Sci. **43** (2009), 408–416.
[18] G. Laporte, Y. Nobert, and S. Taillefer, *Solving a family of multi-depot vehicle routing and location-routing problems*, Transp. Sci. **22** (1988), 161–172.
[19] E. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan, and D. B. Shmoys, *The traveling salesman problem: A guided tour of combinatorial optimization*, John Wiley, Chichester, 1985.
[20] J. MacQueen, *Some methods for classification and analysis of multivariate observations*, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability **1** (1967), 281–297.
[21] A. Novikov, *PyClustering: Data mining library*, J. Open Source Softw. **4** (2019), 1230.
[22] P. Toth and D. Vigo, *An overview of vehicle routing problems*, in *The Vehicle Routing Problem*, P. Toth and D. Vigo, Eds., SIAM, Philadelphia, 2002, 1–26.
[23] A. Van Breedam, *Improvement heuristics for the vehicle routing problem based on simulated annealing*, Eur. J. Oper. Res. **86** (1995), 480–490.

How to cite this article: Fermín Cueto P, Gjeroska I, Solà Vilalta A, Anjos MF. A solution approach for multi-trip vehicle routing problems with time windows, fleet sizing, and depot location. *Networks.* 2021;78:503–522. https://doi.org/10.1002/net.22028

APPENDIX A

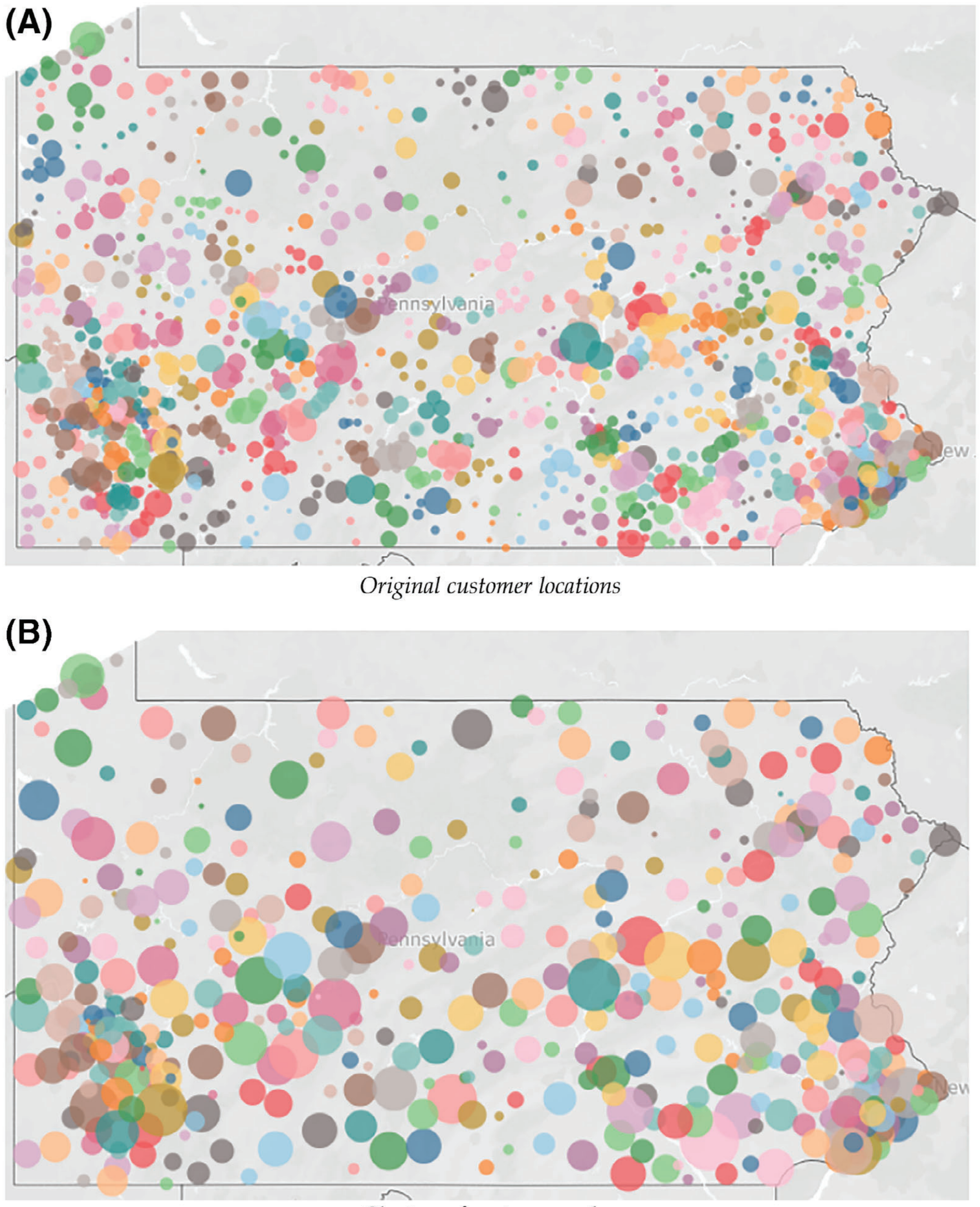

FIGURE A1 Customer node aggregation using *k*-medoids to generate 500 clusters from the initial 1457 customers. The size of the markers indicates the number of days with demand in 2018 for each customer (left) or cluster (right). The color code serves as guidance to identify the cluster to which each customer was assigned

Algorithm 1. Modified FFD heuristic

```
procedure PackRoutesInVehicles(routes, $a_D$, $b_D$, $a_C$, $b_C$)
    initialise list of vehicles with one empty vehicle $k = 1$ and assign $k.gap \leftarrow a_C - a_D$  /* if depot opens at
    6am and customers can be visited from 8am, the initial gap is 2h
    Sort routes in decreasing order by route duration
    for r in routes do
        for k in vehicles do
            $r \leftarrow$ LongArcFirst(r) /* see procedure below
            if $(r.earliest - a_D) \leq k.gap$ then /* if placing route r at the start would reduce
            current starting gap
                Insert route r at the start of schedule of vehicle k
                if IsFeasibleSchedule($k, a_D, b_D, a_C, b_C$) then /* see procedure below
                    $k.gap \leftarrow r.earliest - a_D$  /* update current start gap
                    break  /* this route has been assigned, exit vehicle loop and move to
                    next route
                else
                    Remove route r from vehicle k
                    continue  /* scan next vehicle, this route cannot fit here
            else  /* if this route cannot improve the starting gap
                $r \leftarrow$ LongArcLast(r) /* see procedure below
                if last arc in r longer than last arc of current route in last position then
                    Insert route r at the end of schedule of vehicle k
                else
                    Insert route r in any intermediate position in vehicle k
                if IsFeasibleSchedule($k, a_D, b_D, a_C, b_C$) then
                    break  /* this route has been assigned, exit vehicle loop and move to
                    next route
                else
                    Remove route r from vehicle k
        if r has not been assigned then
            Insert vehicle at the end of list vehicles
            $r \leftarrow$ LongArcFirst(r)
            Insert route r at the start of schedule of vehicle k
            $k.gap \leftarrow r.earliest - a_D$  /* initialise start gap
    return number of elements in list vehicles

procedure LongArcFirst(r)
    Arrange route r in a direction such that it starts with larger between r.first and r.last

procedure LongArcLast(r)
    Arrange route r in a direction such that it ends with larger between r.first and r.last

procedure IsFeasibleSchedule(k, $a_D$, $b_D$, $a_C$, $b_C$)
    Check if the schedule defined by routes currently assigned to vehicle k, in the given order, violates any time windows
```