

# Vehicle Routing with Roaming Delivery Locations

Damián Reyes      Martin Savelsbergh      Alejandro Toriello

H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology

Atlanta, Georgia 30318

*ldrr3 at gatech dot edu, {mwps, atoriello} at isye dot gatech dot edu*

Corresponding author: Damián Reyes

postal address: 765 Ferst Dr NW, Atlanta, GA 30318

email: [ldrr3@gatech.edu](mailto:ldrr3@gatech.edu)

**Keywords:** Vehicle routing; city logistics; last-mile delivery; car-trunk delivery; heuristics; dynamic programming.

# Vehicle Routing with Roaming Delivery Locations

Damián Reyes      Martin Savelsbergh      Alejandro Toriello  
H. Milton Stewart School of Industrial and Systems Engineering  
Georgia Institute of Technology  
Atlanta, Georgia 30318  
ldrr3 at gatech dot edu, {mwps, atoriello} at isye dot gatech dot edu

---

## Abstract

We propose the *vehicle routing problem with roaming delivery locations* (VRPRDL) to model an innovation in last-mile delivery where a customer's order is delivered to the trunk of his car. We develop construction and improvement heuristics for the VRPRDL based on two problem-specific techniques: (1) efficiently optimizing the delivery locations for a fixed customer delivery sequence, and (2) efficiently switching a predecessor's or successor's delivery location during the insertion or deletion of a customer in a route. Furthermore, we conduct an extensive computation study to assess and quantify the benefits of trunk delivery in a variety of settings. The study reveals that a significant reduction in total distance traveled can be achieved, especially when trunk delivery is combined with traditional home delivery, which has both economic and environmental benefits.

---

## 1. Introduction

The growth of e-commerce has caused a significant rise in direct-to-consumer deliveries; this year, for the first time, online purchases have surpassed in-store purchases of retail items (excluding groceries) in the US (Farber, 2016; UPS and comScore, 2016). This poses a major challenge in the so-called “last mile” of the supply chain, the final delivery of goods to the consumer. The last-mile challenge is further exacerbated by the desire for ever faster delivery, to compete with the instant gratification offered by brick-and-mortar stores. Higher volumes of direct-to-consumer e-commerce, tied to inefficient last-mile delivery systems, can lead to a substantial increase in vehicle miles traveled, especially in residential areas, which not only results in high costs, but also in increased emissions, increased congestion, and a decrease in quality of life. The situation is worsened when customers must sign upon delivery (as is the case in most European countries) and multiple visits are made due to missed deliveries.

The negative impacts of an increase in vehicle miles traveled can be mitigated by deploying environmentally-friendly vehicles, and many companies are investigating such options. However, improvements in operational practices can also make a large contribution – for example, Walmart attributes 39% of their efficiency gains during 2005-2015 to improvements in routing (Wal-Mart Stores, Inc., 2016) – and should therefore be explored as well; that is the focus of this paper. The efficiency of last-mile delivery can be improved in at least two ways: by decreasing the distance from the fulfillment center to the delivery locations (which will increase delivery location density) and by reducing the number of missed deliveries. Both of these can be achieved through the use of parcel box deliveries – an increasingly popular option, e.g. United Parcel Service of America, Inc. (2016) – and the use of trunk deliveries, which are the focus of this paper.

By delivering to the trunk of a customer’s car rather than to the customer’s home, delivery may take place (if carefully timed) at a location closer to the fulfillment center, or closer to other delivery locations. Furthermore, there are two main reasons why companies choose not to allow delivery at the door when the customer is not at home - inclement weather and theft risks - and the adoption of trunk delivery greatly reduces both of them, thus eliminating the need for the dreaded “we missed you” notes.

The innovative idea of trunk delivery can be traced back to start-up company Cardrops ([www.cardrops.com](http://www.cardrops.com)), which in 2012 first attempted to tackle the technical challenges through the use of a device with GPS-tracking abilities and control of the trunk lock (Biggs, 2012) installed inside customers vehicles. Shortly afterwards, the security and communication technologies enabling this mode of delivery began to be seamlessly integrated as features of the latest car models – a move part of a broader push by car manufacturers to make physical keys a thing of the past (Davies, 2016). A proof of concept was introduced by Volvo at the 2014 Mobile World Congress (Volvo Cars, 2014), and pilot studies followed suit (Gleyo, 2015):

Holiday shopping will be pleasant for those in Gothenburg, Sweden. Volvo kick-started its own commercially-available In-car Delivery service that aims to liberate consumers who choose to shop online.

With the new service, online shoppers can have the package delivered directly to their cars, even if they are away from it. The whole delivery process becomes feasible with a unique one-time-use digital key that couriers get to unlock the client’s vehicle.

Since then, important car-manufacturing and logistics companies have started partnerships to experiment with the concept, among them Volvo and Urb-it (Korosec, 2016), Daimler, DHL and Amazon (Etherington, 2016), and Audi, DHL and Amazon (Audi, 2015):

“If the Audi owner agrees to the tracking of their automobile for the specific delivery time frame, the DHL driver handling the parcel receives a digital access code for the

trunk of the customer's vehicle. It can be used one time only for a specific period of time and expires as soon as the luggage compartment has been closed again. Similarly, Audi connect easy delivery customers will also be able to send letters and parcels from their own car in the future.

From a methodological perspective, the possibility of trunk deliveries leads to a fundamentally different variant of the well-known *vehicle routing problem* (VRP). The VRP has been extensively studied since its introduction in the early 1950s, and there is a plethora of VRP variants, including the VRP with time windows, the VRP with pickups and deliveries, the dynamic VRP, the VRP with stochastic demands, and the split delivery VRP, to name just a few. In all these problem variants, the customer's delivery location, i.e. the location where the delivery occurs, is given (even if the decision maker may not have perfect information about it). When deliveries are made to the trunk of a customer's car, this constancy disappears, because the customer's car will likely be in different locations during the planning horizon, e.g. at work, at the mall, at church, at the kids' soccer practice, and so on, and a delivery location (and thus delivery time) must be chosen by the service provider. We propose the *VRP with roaming delivery locations* (VRPRDL) as a canonical optimization model to capture this new feature.

The contributions of the research reported in this paper are twofold. First, we introduce an interesting and practically relevant new variant of the VRP, and develop an effective heuristic for its solution. Though partly based on known techniques, the heuristic includes innovations, namely the computationally efficient management of a set of feasible solutions during construction and local search procedures, and the use of an efficient dynamic programming algorithm to optimize parts of the heuristic solution at appropriate times. Second, we conduct an extensive computational study to assess and quantify the benefits of trunk delivery, either as a replacement of home delivery or in conjunction with home delivery. The study reveals that, depending on the geography and the assumptions about daily travel itineraries of customers, the benefits can be significant, in certain settings resulting in a reduction of the total distance traveled of more than 50% (and the concomitant reduction in emissions, congestion, etc.).

The remainder of the paper is organized as follows. We close this section with a brief review of relevant literature. Section 2 then formally introduces the problem and gives an integer programming formulation for it. Section 3 discusses the optimization of a route for a fixed sequence of customers, an important sub-problem. Section 4 proposes various constructive and improvement heuristics, which we compare computationally in Section 5. This section also presents the results of an extensive computational study assessing and quantifying the benefits of trunk delivery. Section 6 closes with some final remarks, while an appendix contains detailed technical material not included in the body of the paper.

### 1.1. Relevant literature

There is a huge and ever-expanding body of literature on vehicle routing and scheduling problems. For conciseness, we provide only a few references for each of the vehicle routing and scheduling variants relevant to our research.

In the canonical vehicle routing problem (VRP), a set of geographically dispersed customers has to be visited to satisfy their demand. The objective is to construct a minimum-cost set of delivery routes serving all customers such that the total demand of the customers served by a single vehicle does not exceed the vehicle capacity (Dantzig and Ramser, 1959; Laporte, 2009). In the vehicle routing problem with time windows (VRPTW), each customer requiring a delivery has a time window during which the delivery can take place (Savelsbergh, 1986; Solomon, 1987; Baldacci et al., 2012; Vidal et al., 2013). (A vehicle is usually allowed to wait at a customer's location until the start of the time window.) In the generalized vehicle routing problem (GVRP), the set of customers is partitioned into clusters and each cluster has a given demand. The objective is to construct a minimum cost set of delivery routes serving one of the

customers in each cluster such that total demand of the customers served by a single vehicle does not exceed the vehicle capacity (Bektaş et al., 2011).

We are aware of only a single paper (Moccia et al., 2012) that considers the generalized vehicle routing problem with time windows (GVRPTW), i.e., the variant that combines the characteristics of the VRPTW and the GVRP. By splitting a single customer in the VRPRDL into multiple customers, one for each of the locations visited in the customer’s geographic profile, we obtain a special case of the GVRPTW. It is a special case, because the time windows of the customers in a cluster do not overlap.

There are other routing models that share some similarities with the VRPRDL. The most important one is the time-dependent vehicle routing problem (TDVRP) where the cost (or travel time) to go from one location to another depends on the departure time (Malandraki and Daskin, 1992; Ichoua et al., 2003; Figliozzi, 2012). In the VRPRDL, the travel time from one customer to the next also depends on the time of departure, as the departure time defines the location where the departure takes place. However, contrary to the TDVRP, the travel time is not uniquely defined by the departure time, because the next customer may be reached in different locations.

## 2. The VRP with roaming delivery locations

The VRP with Roaming Delivery Locations (VRPRDL) can be formally defined as follows. Let  $(N \cup \{0\}, A)$  denote a complete directed graph with node set  $N \cup \{0\}$  and arc set  $A$ , where node 0 represents the depot and  $N$  represents a collection of locations of interest. Each arc  $a \in A$  has an associated travel time  $t_a$  and cost  $w_a$ , both of which satisfy the triangle inequality.  $C$  represents the set of customers that require a delivery during the planning period  $[0, T]$ . The delivery for a customer  $c \in C$  is characterized by a demand quantity  $d_c$  and geographic profile  $N_c \subseteq N$  that specifies where and when a delivery can be made. Specifically, each location  $i \in N_c$  has a non-overlapping time window  $[a_i^c, b_i^c]$  during which the customer’s vehicle is at  $i$ . By duplicating locations, we may assume  $N_c \cap N_{c'} = \emptyset$  for different customers  $c, c'$ , and to simplify notation we also assume  $|N_c| = k$  for all customers  $c$ . Each customer’s time windows naturally imply an ordering of locations  $i_1^c, \dots, i_k^c \in N_c$ ; we assume that  $i_1^c = i_k^c$  represent the customer’s home location and that the time windows satisfy

$$a_1^c = 0; \quad b_k^c = T; \tag{1a}$$

$$a_\ell^c = b_{\ell-1}^c + t_{i_{\ell-1}^c, i_\ell^c}, \quad \ell = 2, \dots, k. \tag{1b}$$

In particular, condition (1b) indicates that when a customer’s vehicle moves from one location to another, it incurs the same travel time as the delivery vehicles and is unavailable during this time; we further explain the reason for this assumption in Section 3 below. A set  $V$  of homogeneous vehicles with capacity  $Q$  is available to make deliveries; vehicles start and end their delivery routes at the depot. The goal is to find a set of delivery routes (a sequence of customer locations) and delivery times such that every customer receives a single delivery during the planning period, the total demand delivered on a delivery route does not exceed  $Q$ , the total travel and waiting time of a delivery route does not exceed  $T$ , and the total cost is minimized. The traditional VRP is the special case in which  $|N_c| = 1$ , i.e., the delivery location is fixed and does not change during the planning horizon.

### 2.1. Integer programming formulation

We next introduce an arc-based formulation of the VRPRDL as a mixed integer program. The formulation uses modeling techniques from routing problems with time windows, see, e.g. Desrochers et al. (1988). We use the following decision variables:

$x_{ij} \in \{0, 1\}$  : indicates whether a vehicle travels from location  $i$  to  $j$ , for  $i, j \in N \cup \{0\}$

$\tau_c \in [0, T]$  : time of departure after service to customer  $c \in C$  at any of its locations  $N_c$   
 $y_c \in [0, Q]$  : cargo remaining on vehicle after service to customer  $c \in C$ .

The formulation is then given by

$$\min_{x, y, \tau} \sum_{i, j \in N \cup \{0\}} w_{ij} x_{ij} \quad (2a)$$

$$\text{s.t.} \quad \sum_{j \in N \cup \{0\} \setminus \{i\}} x_{ij} = \sum_{j \in N \cup \{0\} \setminus \{i\}} x_{ji}, \quad \forall i \in N \cup \{0\} \quad (2b)$$

$$\sum_{i \in N_c} \sum_{j \in N \cup \{0\} \setminus \{i\}} x_{ij} = 1, \quad \forall c \in C \quad (2c)$$

$$\sum_{i \in N} x_{0i} \leq |V|, \quad (2d)$$

$$\sum_{i \in N_c} a_i^c \sum_{j \in N \cup \{0\} \setminus \{i\}} x_{ij} \leq \tau_c \leq \sum_{i \in N_c} b_i^c \sum_{j \in N \cup \{0\} \setminus \{i\}} x_{ij}, \quad \forall c \in C \quad (2e)$$

$$\tau_c + \sum_{i \in N_c} \sum_{j \in N_{c'}} t_{ij} x_{ij} \leq \tau_{c'} + T \left( 1 - \sum_{i \in N_c} \sum_{j \in N_{c'}} x_{ij} \right), \quad \forall c \in C \cup \{0\}, c' \in C \setminus \{c\} \quad (2f)$$

$$0 \leq y_c \leq Q - d_c, \quad \forall c \in C \quad (2g)$$

$$y_c + Q \left( 1 - \sum_{i \in N_c} \sum_{j \in N_{c'}} x_{ij} \right) \geq d_{c'} + y_{c'}, \quad \forall c \in C \cup \{0\}, c' \in C \setminus \{c\} \quad (2h)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in N \cup \{0\}. \quad (2i)$$

In the formulation we use the depot 0 both as a location and as a dummy customer, so that  $N_0 = \{0\}$ . Similarly, we can take  $\tau_0 = 0$ ,  $y_0 = Q$ , and  $[a_0, b_0] = [0, T]$ . Constraint (2b) describes flow conservation for every location, and (2c) enforces exactly one visit per customer. Similarly, (2d) limits the total number of routes to the number of vehicles. Constraint (2e) enforces the time windows, while (2f) ensures that departure and travel times are consistent. Constraints (2h) and (2g) serve a similar function for the cargo variables.

### 3. Optimizing the delivery cost for a fixed customer sequence

The fact that a customer has multiple delivery locations during the planning horizon implies that specifying customer delivery sequences is no longer sufficient to specify a solution. In traditional vehicle routing settings, for a given sequence it is straightforward to ascertain that the delivery route is feasible and to determine an associated minimum cost. (In almost all VRP variants, for a given route it is optimal to deliver to a customer as early as possible.)

In the VRPRDL, a set of customer delivery sequences does not automatically provide a set of delivery routes, because a customer can be visited at one of several locations. Thus, a core optimization problem in the context of the VRPRDL is to determine optimal customer delivery times and locations for a given sequence. Throughout this section we assume a given customer sequence  $(1, \dots, m)$  with  $d_1 + \dots + d_m \leq Q$ .

We solve the problem using forward dynamic programming (DP) on a time-expanded network with nodes of the form  $(c, \tau, i)$ , where  $c = 0, \dots, m + 1$  represents the current customer (and  $0, m + 1$  are, respectively, the start and end of the route),  $\tau \in [0, T]$  is the departure time from customer  $c$ , and  $i \in N_c$  is the location where the delivery to  $c$  took place; the latter can be inferred from  $\tau$  and  $c$ 's time windows, but we include it to simplify our exposition. The recursion is then given by

$$z(0, 0, 0) = 0 \quad (3a)$$

$$z(c, \tau, j) = \min \left\{ z(c-1, \tau', i) + w_{ij} : \begin{array}{l} i \in N_{c-1}, \quad a_i^{c-1} \leq \tau' \leq \min\{b_i^{c-1}, \tau - t_{ij}\}, \\ c = 1, \dots, m+1, \quad j \in N_c, \quad \tau \in [a_j^c, b_j^c]. \end{array} \right\}, \quad (3b)$$

The quantity  $z(c, \tau, j)$  is the minimum cost of a partial route  $(0, \dots, c)$  that delivers to customer  $c$  at location  $j$  at time  $\tau$ , which implies that the minimum cost route is found by taking  $\min\{z(m+1, \tau, 0) : \tau \in [0, T]\}$ . The recursion (3b) states that the minimum cost of a delivery to customer  $c$  at location  $j$  at time  $\tau$  is obtained as the sum of the minimum cost of a delivery to the previous customer in the sequence, i.e.,  $c-1$ , at location  $i$  at time  $\tau'$  in the time interval  $[a_i^{c-1}, \min\{b_i^{c-1}, \tau - t_{ij}\}]$ , i.e., the feasible departure times at location  $i$  that arrive at location  $j$  at or before time  $\tau$  and the cost of travel from location  $i$  to location  $j$ . These quantities, as defined, range over all possible values of  $\tau \in [0, T]$  at every step, which may be impractical even if all times are integer-valued, and may be intractable otherwise. However, we next show that the optimal route for the fixed sequence can be calculated efficiently.

**Theorem 1.** *For a fixed customer sequence  $(1, \dots, m)$ , the optimal delivery route can be calculated in  $O(k^2 m^2)$  time.*

*Proof.* To prove the theorem we must show that the number of nodes the algorithm explores does not grow excessively. Beginning recursion (3) at  $(0, 0, 0)$ , the algorithm only evaluates nodes that are reachable from a previously evaluated node and non-dominated in terms of time and cost; the latter condition implies that the vehicle only waits at a location if it arrives before the time window.

We define a node to be *regular* if it is of the form  $(c, a_i^c, i)$ , i.e. the vehicle delivers at the earliest moment in a location's time window, and *irregular* otherwise. From any node the algorithm is currently evaluating, (1b) and the triangle inequality guarantee that at most one of the non-dominated reachable nodes will be irregular: If location  $i$  is reachable within its time window, by condition (1b) the vehicle could "follow" the customer from there to any later location and make the delivery at the start of that window. Figure 1 shows an illustration of (the first layers of) a time-expanded network, with regular and irregular nodes.

The proof is completed by bounding the number of regular and irregular nodes the algorithm explores: Customer 1 has at most  $k$  nodes (with one possibly being irregular); these nodes then generate at most  $k$  irregular nodes for customer 2, which are added to its (no more than)  $k$  regular nodes. By induction, when customer  $c$  in the sequence is reached, no more than  $ck$  nodes will be evaluated, and thus we conclude that a total of  $O(km^2)$  nodes and  $O(k^2 m^2)$  arcs will be evaluated by the algorithm.  $\square$

The proof of Theorem 1 implies that condition (1b) can be relaxed to require that the time between windows be at least the travel time between the corresponding locations. However, the argument breaks down if this time is shorter, because we can no longer guarantee that only one irregular node is generated at every evaluation.

The algorithm can be sped up heuristically by eliminating dominated irregular nodes. For any customer  $c$  and location  $i$ , if  $z(c, \tau, i) \leq z(c, \tau', i)$  and  $\tau < \tau'$ , the latter node can be eliminated without evaluation, since any nodes it can reach are also reachable from the former at equal or lower cost.

## 4. Heuristics

The VRPRDL generalizes the VRP and is a difficult optimization problem from both a theoretical and practical perspective. As with the VRP, one option to produce high-quality solutions is via computationally efficient heuristic methods.

In the case of the VRPRDL, these methods can harness the flexibility afforded by the multiple delivery locations available per customer. Specifically, the heuristics we propose rely

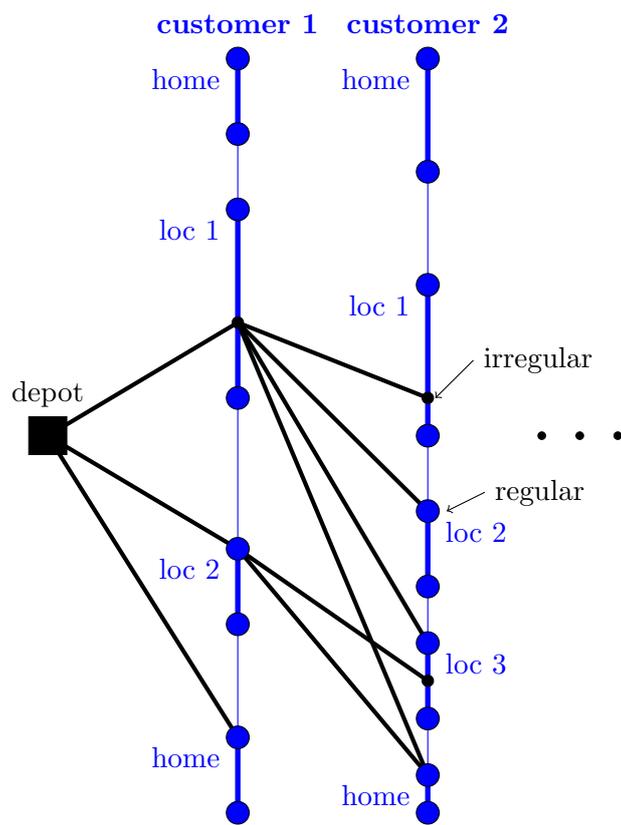


Figure 1: Illustration of the time-expanded network used in the dynamic program (showing the first two customers only).

on applying many *enhanced* insertion and deletion operations to a given route; if we fix each customer’s delivery location on the route, these operations are identical to their analogues in a classical VRP or other routing model with time windows. However, we can increase the operations’ flexibility and the resulting potential of heuristics to produce high-quality solutions by considering also the shifting of customers to different locations (and thus a shift in the corresponding time windows) within the insertion and deletion procedure. To do so, given an incumbent route, in addition to the route itself and its time window feasibility information, we maintain a collection of alternate routes that differ from the incumbent in exactly one customer’s delivery location; similar ideas have been used in other constrained vehicle routing contexts, e.g. Campbell and Savelsbergh (2004).

Assume again that we have a fixed sequence of customers  $(1, \dots, m)$  with  $d_1 + \dots + d_m \leq Q$  and an incumbent feasible route  $r = (i^1, \dots, i^m)$  serving these customers in this order, where  $i^c \in N_c$  for each  $c = 1, \dots, m$ . To implement our enhanced insertion operations when time windows are present, it is necessary to calculate the “effective” time windows for each customer location, i.e. a window  $[\alpha_{i^c}^c, \beta_{i^c}^c]$  specifying the earliest and latest times the customer can feasibly be served by a vehicle following this route. These windows are calculated using the recursions

$$\alpha_{i^1}^1 = a_{i^1}^1, \quad \alpha_{i^c}^c = \max\{a_{i^c}^c, \alpha_{i^{c-1}}^{c-1} + t_{i^{c-1}, i^c}\}, \quad c = 2, \dots, m; \quad (4a)$$

$$\beta_{i^m}^m = b_{i^m}^m, \quad \beta_{i^c}^c = \min\{b_{i^c}^c, \beta_{i^{c+1}}^{c+1} - t_{i^c, i^{c+1}}\}, \quad c = 1, \dots, m-1. \quad (4b)$$

In addition, for each customer  $c = 1, \dots, m$  we also maintain similar time windows for every location  $i \in N_c \setminus \{i^c\}$  defined as

$$\alpha_i^c = \max\{a_i^c, \alpha_{i^{c-1}}^{c-1} + t_{i^{c-1}, i}\}, \quad \beta_i^c = \min\{b_i^c, \beta_{i^{c+1}}^{c+1} - t_{i, i^{c+1}}\}. \quad (4c)$$

Assuming  $\alpha_i^c \leq \beta_i^c$ , this window represents the earliest and latest possible delivery to  $c$  if the delivery occurs at the alternate location  $i$ , with the remainder of the route unchanged. With these effective time windows available, we can verify whether a new customer  $\hat{c}$  can be inserted at location  $\hat{i} \in N_{\hat{c}}$  between customers  $c-1$  and  $c$  while simultaneously switching  $c-1$  to delivery location  $i \in N_{c-1}$  and  $c$  to delivery location  $j \in N_c$  if

$$\max\{a_{\hat{i}}^{\hat{c}}, \alpha_i^{c-1} + t_{i, \hat{i}}\} \leq \min\{b_{\hat{i}}^{\hat{c}}, \beta_j^c - t_{\hat{i}, j}\}; \quad (4d)$$

the check can be carried out in constant time. If the condition is satisfied, the two quantities become the new location’s effective time window, and we redefine the  $\alpha$  values for  $c, \dots, m$  and  $\beta$  values for  $1, \dots, c-1$  accordingly in  $O(km)$  time. The new route would have  $\hat{c}$  in position  $c$ , with subsequent customers shifted forward by one position. This also means we can optimize the insertion of a customer at any position in the route, to any of the customer’s delivery locations, while also changing its predecessor’s and successor’s locations in  $O(k^3m)$  time. Similar but more restricted operations can reduce the complexity; for example, if the insertion changes only one of the predecessor’s or successor’s location, the operation only takes  $O(k^2m)$  time. A similar but simpler check and update can be implemented to enhance the deletion of a customer from a route. Figure 2 and 3 illustrate the potential benefits of these enhanced insertion and deletion operations.

These basic (enhanced) insert and delete operations with simultaneous predecessor and/or successor location change can be embedded in more sophisticated heuristics and meta-heuristics. We discuss one possible pair of construction and improvement heuristics next.

#### 4.1. Construction heuristic

The construction heuristic we propose for the VRPRDL is inspired by the family of *greedy randomized adaptive search procedures* (Feo and Resende, 1995). The procedure repeats a

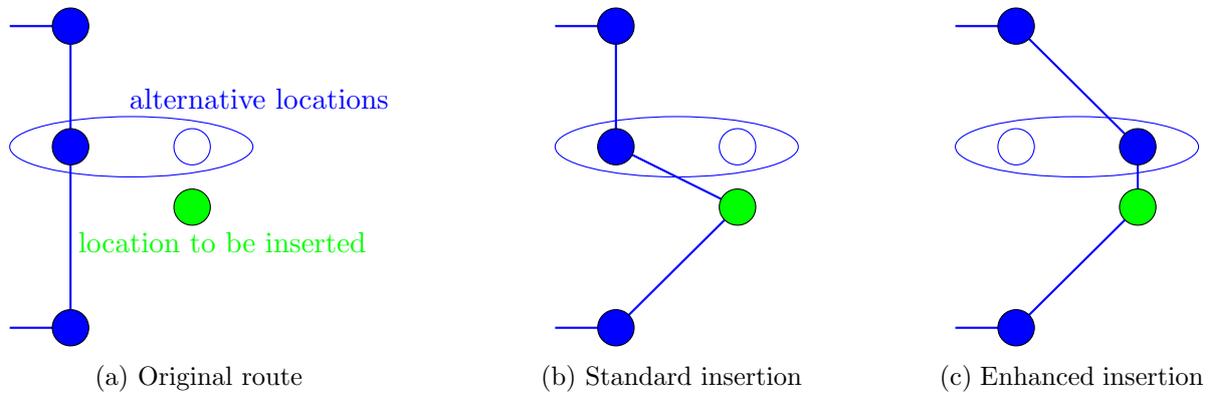


Figure 2: Potential benefit of an enhanced insertion.

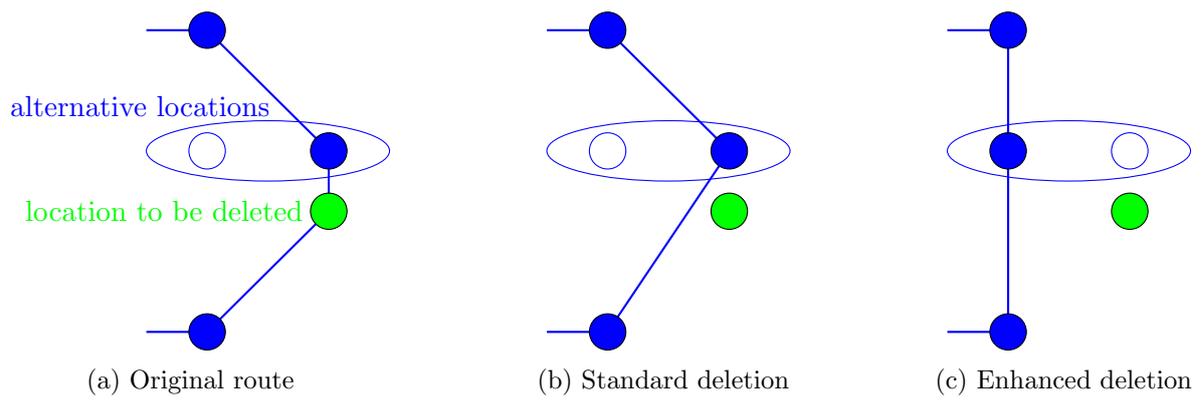


Figure 3: Potential benefit of an enhanced deletion.

randomized construction procedure  $N$  times, choosing the best solution. Each iteration produces a feasible solution by performing a basic insertion one customer at a time.

At each step in the construction, the next customer to be inserted is picked randomly from a restricted candidate list defined in the following manner:

1. For each customer  $c$  not yet assigned to a route in the solution, for each location  $i \in N_c$ , and for each route  $r$  among the routes in the solution (including the empty route), evaluate the cost change induced by the basic insertion operation at every point in the route (which may involve changing the predecessor or successor location).
2. Rank the options by nondecreasing cost and keep the  $K$  most profitable.

In our experiments, we used the parameters  $N = 100$  and  $K = 2$ ; we chose this construction heuristic with these parameters after comparing it in preliminary experiments against other methods, such as a simpler greedy rule and a carousel-greedy heuristic (Cerrone et al., 2014). It also consistently outperformed the best solution found by Gurobi after two hours using the formulation (2). Algorithm 4.1 outlines the construction heuristic in pseudo-code.

---

**Algorithm 4.1: CONSTRUCTION HEURISTIC**

---

**Input:** Solution pool size  $N$ . Restricted candidate list size  $K$ .

**Output:** Solution  $S$  consisting of feasible routes serving all customers.

// Provided that the instance is feasible, this procedure returns a feasible solution with the least cost among  $N$  solutions constructed using a  $K$ -randomized greedy parallel insertion heuristic.

```

Pool ← ∅
for count ← 1 to N do
    R ← ∅ // solution to be constructed
    Unserved ← C
    while Unserved ≠ ∅ do
        r, j, c, i ← RANDGREEDYPARALLELINSERTION(R, Unserved, K)
        // c to be visited at location i as jth customer in route r
        if r = ∅ then
            return ∅ // unable to construct feasible solution
        end
        else if r ∉ R then
            R ← append r // include new route in solution
        end
        INSERT(r, j, c, i)
        Unserved ← Unserved \ c
    end
    Pool ← append R
end
S ← arg min{cost(R) | R ∈ Pool}
return S

```

---

#### 4.2. Improvement heuristic

The improvement heuristic we propose implements a variable neighborhood search using the destroy and recreate paradigm; see e.g. Pisinger and Røpke (2007). Each iteration of a

destroy-recreate routine is composed of two phases. In the first phase, a feasible solution is “destroyed” by sequentially executing  $L$  basic delete operations. In the second phase, a new feasible solution is “recreated” through a sequence of basic insertion operations that reassign routes to the  $L$  deleted customers, respecting a first-deleted first-reinserted rule.

Different destroy-recreate neighborhoods can be induced by implementing different selection rules to determine the specific customer, location, and route on which an operation acts. We tested the following variants in our heuristic:

1. Greedy destroy and greedy recreate (GDGR).
2. Randomized greedy destroy and randomized greedy recreate (rGDrGR).
3. Random destroy and greedy recreate rule (RDGR).

At each step of a destruction phase, we consider a candidate deletion for every customer-route assignment in the (partial) solution. Among all these options, the greedy destroy rule executes the operation that yields the largest savings, whereas the  $K$ -randomized greedy destroy rule picks an option randomly among the  $K$  deletions with the largest savings. Finally, the random destroy rule chooses the customer to delete completely at random. Notice that the first and last rules are extremes of the randomized greedy rule.

We employ a similar rationale in the recreation phase, but the set of candidate insertions has a different form. First, because of the first-deleted first-reinserted rule, the customer  $c$  to reinsert is fixed. Second,  $c$  has no assigned route, position in the route, nor predetermined delivery location. Finally, inserting  $c$  into an “empty” new route may be the most appealing alternative. Thus, there is a candidate insertion for every delivery location  $i$  of customer  $c$ , for each route  $r$  in the solution (plus the empty route), and for every position in the route.

In our implementation, to avoid local optima our heuristic alternates between rGDrGR and RDGR, switching whenever an improvement is not found after 10 iterations. During a switch, we employ a greedy destroy and then run the DP recursion (3) on each route in the partial solution before the next recreate phase. We run the procedure for a total of 2000 iterations, but after 1000 iterations we perform a one-time diversification, where the most expensive route is divided into two routes of approximately the same number of customers. We set the size of the restricted candidate list based on the number of customers,  $K^{del} = \lceil 0.2n \rceil$  and  $K^{ins} = \lceil 0.05n \rceil$ . All of the parameter values were chosen based on initial tuning tests. Algorithm 4.2 details the improvement heuristic.

## 5. Computational study

The goal of our computational study is to demonstrate that (1) the algorithmic ideas presented form a solid foundation on which to build the enabling technologies to support trunk deliveries, and, more importantly, that (2) a trunk delivery system can have substantial economic and environmental benefits. To substantiate these claims, we present the results of a series of computational experiments. First, we demonstrate the effectiveness of our heuristics on a set of instances that were generated using as few assumptions as possible about customers’ geographic profiles. Second, we focus on the value of trunk delivery systems by studying a set of instances generated to resemble practical situations. In all our experiments, total distance traveled is the relevant cost metric being minimized.

### 5.1. Instances

Two sets of randomly generated instances have been used in our computational experiment: a set of “general” instances and a set of “realistic” instances resembling situations likely to be encountered in real-life trunk delivery operations.

A general instance is characterized by a planning horizon,  $T$ , a number of customers,  $n$ , and for each of customer  $c$ , a demand,  $d_c$ , a home location,  $(x_c, y_c)$ , a sequence of  $m_c$  roaming

---

**Algorithm 4.2: IMPROVEMENT HEURISTIC**

---

**Input:**

$S$ , a feasible solution

$I$ , number of iterations

$L$ , number of customers to be deleted and reinserted at each iteration

$K^{del}, K^{ins}$ , size of restricted candidate lists

$switch$ , number unsuccessful attempts before switching destroy and recreate neighborhoods

$split$ , iteration at which the costliest route is broken into two

**Output:**  $S$ , a feasible solution

$lastImprovement \leftarrow 0$

$lastSwitch \leftarrow 0$

$L \leftarrow L^1$

**for**  $iter \leftarrow 1$  **to**  $I$  **do**

**if**  $iter = split$  **then**

$S \leftarrow \text{SPLITCOSTLIESTROUTE}(S)$

**end**

$R \leftarrow \text{DESTROYRECREATE}(S, L, K^{del}, K^{ins})$

**if**  $cost(R) \geq cost(S)$  **then**

**if**  $iter - lastImprovement \geq switch$  **then**

$R, Unserved \leftarrow \text{DESTROY}(S, L, 1)$

$R \leftarrow \text{DP}(R)$

$S \leftarrow \text{RECREATE}(R, Unserved, 1)$

$lastSwitch \leftarrow iter$

$\text{SWITCHNEIGHBORHOOD}()$

**end**

**end**

**else**

$S \leftarrow R$

$lastImprovement \leftarrow iter$

**end**

**end**

**return**  $S$

---

Table 1: Coordinates of the centers of the work clusters.

Downtown	0	0
Buckhead	0	15
Airport	0	-15
Alpharetta	0	30
Marietta	-25	15
Doraville	25	15
Decatur	25	0
Douglasville	-25	0

locations, in the order in which these locations are visited, and, finally, the time spent at every location (as a fraction of the planning horizon). The home location of each customer is reachable by an out-and-back tour from the depot, located at the center of the region,  $(0, 0)$ . The roaming locations of each customer are centered around the home location and can all be visited during the planning period in consecutive out-and-back trips from home (consequently, they can be visited in sequence, by the triangle inequality). The time *not* consumed by traveling is partitioned into  $m_c + 1$  pieces of uniformly random lengths and linked to each location of the sequence (the home has two time windows, one at the beginning and one at the end of the planning period). A detailed description of the instance generator can be found in Algorithm A.1 in the appendix. It has two important control parameters: the maximum number of roaming delivery locations visited by a customer and the maximum distance of a roaming delivery location from the home location. We have generated a set of 40 instances of increasing size and complexity with the number of customers ranging from 15 to 120, number of locations visited during the planning period ranging from 1 to 5 (if only one location is visited, it implies the customer stays at home the whole time), and a planning period of 12 hours.

A realistic instance, too, is characterized by a planning horizon,  $T$ , a number of customers,  $n$ , and for each customer  $c$ , a demand,  $d_c$ , and a home location,  $(x_c, y_c)$ . However, the number of roaming locations is determined by the customer type, which can be one of three: at home only; at home and at work; and at home, at work, and somewhere else after work (e.g., at a shopping center, at a gym, or at a child’s soccer practice). In all instances, the three types get 10%, 40%, and 50% of the customers, respectively. Work locations are in pre-defined clusters. Our instances are inspired by the geography of Atlanta and have eight work clusters. The coordinates of the centers of these work clusters (in minutes, driving from downtown) can be found in Table 1. The planning period is 14 hours long, extending from 6am to 8pm. Customers that work each have one of three schedule types: part-time, which implies they work 4 hours starting either at 8am or at noon; almost full-time, which implies they work between 4 and 7 hours starting between 8 and 10am; and full-time, which implies they work between 7 and 8 hours starting between 8 and 9am. Everyone is assumed to arrive at work exactly at the time they start work. Of the customers that work, 10% are part-time, 10% almost full-time, and 80% are full-time. Customers that go somewhere else after work visit one of 30 locations spread across the region; specifically, they visit the after-work location that is closest to the home location. The time spent at the after-work location is randomly chosen, but is less than 50% of the time between the end of work and the end of the day. A detailed description of the instance generator can be found in Algorithm A.2 in the appendix.

## 5.2. Algorithm performance

Because we are the first to explore the VRPRDL, there are no existing solution approaches to benchmark with, so we compare the quality of the schedules produced by our heuristic, which we denote by HEURRDL, to the quality of the schedules produced when we solve the integer

program (2) using the commercial integer programming software Gurobi 5.6 with a time limit of two hours. To ensure as fair a comparison as possible, we set the Gurobi search focus on feasibility ( $MIPFocus = 1$ ) and provide the solution produced by our construction heuristic as a warm start. The parameter values used for the construction heuristic are  $N = 100$  and  $K = 2$ , and the parameters used for the improvement heuristic are  $I = 2000$ ,  $K^{del} = \lceil 0.2|C| \rceil$ ,  $K^{ins} = \lceil 0.05|C| \rceil$ ,  $switch = 10$ , and  $split = 1000$ .

The results for the 40 general instances can be found in Table 2, where we report the instance identifier, the number of customers in the instance, the total number of locations in the instance, the average percentage of time that a customer is available to receive deliveries (*i.e.* is not driving around), the cost of the schedule produced by the construction heuristic,  $c_{init}$ , the cost of the schedule after it has been improved,  $c_{heur}$ , the cost of the schedule produced by the integer programming solver,  $c_{IP}$ , and the relative difference in schedule cost,  $(c_{IP} - c_{heur})/c_{heur}$  (as a percentage). The instances for which Gurobi was able to prove optimality are labeled with a check mark.

When provided with the initial solution produced by our construction heuristic, the integer programming solver was able to solve all instances with  $n = 15$ , all but one instance with  $n = 20$ , and two instances with  $n = 30$  to optimality in two hours, but was unable to prove optimality for any of the instances with 60 or 120 customers. For the small instances ( $n \leq 30$ ), HEURRDL produced schedules of comparable quality to those produced by the integer programming solver, and in all but one of the larger instances ( $n \geq 60$ ) HEURRDL produced schedules of much better quality than the schedules produced by Gurobi; for the largest instances ( $n = 120$ ) the cost is often reduced by more than 20%. We conducted a paired t-test on the difference of means of the solutions produced by HEURRDL and Gurobi for each group of instances of similar size, and found that the advantage in favor of HEURRDL is statistically significant at the 5% level for the groups of large instances and for the overall set of general instances.

In terms of the various components of the heuristic, including our problem-specific innovations, we observed the following behavior. The RDGR neighborhood (random deletions and greedy insertions) was far more effective in finding improvements than the rGDrGR neighborhood (randomized greedy deletions and randomized greedy insertions). In fact, for the larger instances ( $n \geq 60$ ) no improved schedules were found using the rGDrGR neighborhood. On the other hand, for larger instances, the dynamic program, invoked during a switch of neighborhoods, often finds improvements. Furthermore, the flexibility to switch to an alternative delivery location is exploited regularly, especially on smaller instances ( $n \leq 30$ ). The appendix includes detailed statistics on these experiments.

The instance characteristic most likely to impact the performance of an integer programming approach is the number of roaming delivery locations per customer. A second factor that may influence performance is the tightness of time windows (for which the average time available for delivery is a proxy). To quantify this impact, we next compare the quality of the solution produced by HEURRDL to the quality of the solution produced by Gurobi when the number of roaming delivery locations per customer changes. To this end, we construct a separate set of 25 general instances with 60 customers each, in which all customers have the same number  $m$  of roaming delivery locations for  $m = 2, 3, 4, 5$ , and 6 (5 instances for each class). The results, summarized in Table 3, show that if customers have a large number of roaming delivery locations ( $m \geq 5$ ), the quality of the solutions produced by HEURRDL is significantly better than the quality of the solutions produced by Gurobi. It appears that when the time available to make a delivery at customers decreases and the number of locations where such a delivery can be made increases, the linear programming relaxation becomes weaker and the integer programming solver struggles, while HEURRDL continues to find local improvements.

The correlation between number of locations and time available to receive deliveries is quite strong ( $\rho = -0.88$ ) in this sample, which means that a linear regression model including both

Table 2: Performance analysis on general instances.

Instance	# Cust. (n)	# Loc.	Avg. % time available for delivery	$c_{init}$	$c_{heur}$	$c_{IP}$	$\frac{(c_{IP}-c_{heur})}{c_{heur}}\%$
1	15	51	64.2	2463	2128	2128 <sup>✓</sup>	0.0
2	15	53	69.9	2244	2007	1984 <sup>✓</sup>	-1.1
3	15	53	71.3	4023	3661	3661 <sup>✓</sup>	0.0
4	15	58	69.1	2746	2582	2572 <sup>✓</sup>	-0.4
5	15	63	63.1	2101	1802	1802 <sup>✓</sup>	0.0
			Group mean	2715.4	2436.0	2429.4	-0.3
6	20	64	69.5	3749	3374	3374 <sup>✓</sup>	0.0
7	20	67	69.9	3042	2588	2588 <sup>✓</sup>	0.0
8	20	69	67.7	2995	2489	2310	-7.2
9	20	77	53.9	2761	2536	2521 <sup>✓</sup>	-0.6
10	20	81	60.0	3242	3196	2913 <sup>✓</sup>	-8.9
			Group mean	3157.8	2836.6	2741.2	-3.4
11	30	76	78.1	4325	3659	3685	0.7
12	30	99	65.0	4975	4173	4173	0.0
13	30	104	64.2	4608	3849	3849	0.0
14	30	107	61.1	4617	3668	3659 <sup>✓</sup>	-0.2
15	30	108	63.6	3102	2548	2543	-0.2
16	30	114	56.4	5382	4695	4656	-0.8
17	30	119	57.0	4315	3507	3492	-0.4
18	30	120	61.5	4317	3877	3875	-0.1
19	30	125	56.7	4082	3397	3390	-0.2
20	30	131	53.0	4842	3939	3936 <sup>✓</sup>	-0.1
			Group mean	4456.5	3731.2	3725.8	-0.1
21	60	209	64.2	8370	6049	6121	1.2
22	60	214	58.9	7878	5873	6348	8.1
23	60	220	57.8	9843	8391	9029	7.6
24	60	226	61.0	9567	7670	7777	1.4
25	60	226	57.6	10354	9218	9093	-1.4
26	60	227	60.1	9425	8057	8058	0.0
27	60	230	63.9	9336	7032	7237	2.9
28	60	235	60.0	8260	6434	7607	18.2
29	60	236	56.0	10782	8971	10591	18.1
30	60	239	60.7	10048	8428	9853	16.9
			Group mean	9386.3	7612.3	8171.4	7.3*
31	120	423	62.9	17206	13414	16816	25.4
32	120	423	62.3	15367	11434	15244	33.3
33	120	429	59.7	16054	12987	15971	23.0
34	120	442	59.9	14408	11379	13434	18.1
35	120	452	56.6	14536	11713	14067	20.1
36	120	456	59.7	14323	11374	13596	19.5
37	120	462	60.5	13928	10516	13792	31.2
38	120	463	55.9	13455	11045	13312	20.5
39	120	468	56.0	13926	10115	13549	33.9
40	120	472	57.6	14005	10492	13892	32.4
			Group mean	14720.8	11446.9	14367.3	25.5*
			Overall Mean	7875.1	6356.7	7212.5	13.5*

<sup>✓</sup> optimal value.

\* significant difference at the 0.05 level (paired t-test on  $c_{heur}$  and  $c_{IP}$  means).

Table 3: Performance analysis on general instances with controlled number of locations.

Instance	# Cust.	# Loc. per Cust.	Avg. % time available for delivery	$c_{init}$	$c_{heur}$	$c_{IP}$	$\frac{(c_{IP}-c_{heur})}{c_{heur}}\%$
41	60	2	51.4	9178	8125	7877	-3.1
42			57.5	10650	9482	9378	-1.1
43			58.1	11754	10294	10423	1.3
44			63.7	8170	6541	6540	0.0
45			57.9	9088	8096	8079	-0.2
			Group mean	9768.0	8507.6	8459.4	-0.6
46		3	45.5	8652	7308	7264	-0.6
47			47.9	6368	5383	5038	-6.4
48			50.2	9325	7439	7326	-1.5
49			48.2	9807	8039	7670	-4.6
50			48.1	7570	6506	6187	-4.9
			Group mean	8344.4	6935.0	6697.0	-3.4*
51		4	41.6	7930	6561	6508	-0.8
52			46.9	8581	6959	7541	8.4
53			41.4	7928	6509	6426	-1.3
54			42.7	7319	6184	5981	-3.3
55			43.0	7799	6675	6024	-9.8
			Group mean	7911.4	6577.6	6496.0	-1.2
56		5	38.9	9711	8857	8602	-2.9
57			44.4	8455	6952	6980	0.4
58			39.9	7486	5865	7126	21.5
59			41.7	8918	7374	7867	6.7
60			40.8	8745	6953	8716	25.4
			Group mean	8663.0	7200.2	7858.2	9.1
61		6	38.3	8047	6567	7977	21.5
62			36.6	8228	6752	8123	20.3
63			40.3	8488	7262	7971	9.8
64			42.2	9760	8103	9630	18.8
65			39.0	7556	6285	7435	18.3
			Group mean	8415.8	6993.8	8227.2	17.6*
Overall Mean				8620.52	7242.84	7547.56	4.2*

✓ optimal value.

\* significant difference at the 0.05 level (paired t-test on  $c_{heur}$  and  $c_{IP}$  means).

predictors may be ill-conditioned. For this reason, we report the fit of the data onto a simple linear regression model,  $\log(c_{IP}/c_{heur}) = \beta_0 + \beta_1 m + \varepsilon$ : such model yields an adjusted coefficient of determination  $Adj.R^2 = 0.47$ , and regression estimates  $\beta_0 = -0.064$  (p-value= 0.002),  $\beta_1 = 0.020$  (p-value= 0.0001). A similar simple linear regression with time available for deliveries as predictor results in a model with much lower significance ( $Adj.R^2 = 0.19$ ; slope coefficient  $-0.28$ , with p-value= 0.017), which supports our claim that, by far, the number of locations is the most important predictor of differences in performance between our heuristic and the IP solver.

Finally, we assess the performance of HEURRDL on the 40 realistic instances; the results can be found in Table 4. Again, HEURRDL does significantly better (in both the practical and statistical senses), as the cost of the solutions produced by Gurobi is on average 14% higher than the cost of the heuristic solutions.

### 5.3. The benefits of trunk delivery

To analyze the potential benefits of a trunk delivery system compared to a traditional home delivery system, we conducted computational experiments with both the general and the realistic instances.

First, we focus on the general instances, but use four different incarnations of each instance. In addition to the original instance, we consider three additional variations. In each of these

Table 4: Performance analysis on realistic instances.

Instance	# Cust.	# Loc.	Avg. % time available for delivery	$c_{init}$	$c_{heur}$	$c_{IP}$	$\frac{(c_{IP}-c_{heur})}{c_{heur}}\%$
r1a	60	200	89.4	514	443	487	9.9
r2a			88.9	579	511	553	8.2
r3a			89.0	723	597	694	16.2
r4a			89.3	578	467	527	12.8
r5a			88.5	545	478	537	12.3
r6a			88.0	604	490	604	23.3
r7a			88.4	579	466	579	24.2
r8a			89.1	541	413	444	7.5
r9a			88.2	628	499	530	6.2
r10a			88.9	521	450	512	13.8
			Group mean	581.2	481.4	546.7	13.6*
r11a	90	299	88.2	889	805	812	0.9
r12a			88.7	749	636	728	14.5
r13a			87.6	805	647	801	23.8
r14a			88.2	601	565	541	-4.2
r15a			88.5	759	659	759	15.2
r16a			88.4	780	685	777	13.4
r17a			89.2	830	717	802	11.9
r18a			88.0	683	592	638	7.8
r19a			88.2	822	687	802	16.7
r20a			88.3	709	602	659	9.5
			Group mean	762.7	659.5	731.9	11.0*
r21a	120	398	88.2	982	830	980	18.1
r22a			88.3	1025	855	1018	19.1
r23a			89.5	1008	903	995	10.2
r24a			88.1	1019	840	990	17.9
r25a			88.5	1029	875	968	10.6
r26a			88.2	1026	907	1009	11.2
r27a			88.3	983	864	982	13.7
r28a			88.6	1006	841	1006	19.6
r29a			88.7	979	815	966	18.5
r30a			87.9	936	750	909	21.2
			Group mean	999.3	848.0	982.3	15.8*
r31a	150	497	88.5	1078	932	1054	13.1
r32a			88.1	1154	949	1130	19.1
r33a			88.8	1205	976	1165	19.4
r34a			88.5	1042	878	1034	17.8
r35a			88.2	1082	962	1081	12.4
r36a			88.4	1099	973	1080	11.0
r37a			88.3	1138	905	1082	19.6
r38a			88.9	1025	938	977	4.2
r39a			88.1	1085	880	1067	21.3
r40a			88.8	1066	922	1066	15.6
			Group mean	1097.4	931.5	1073.6	15.3*
Overall mean				860.15	730.1	833.625	14.2*

✓ optimal value.

\* significant difference at the 0.05 level (paired t-test on  $c_{heur}$  and  $c_{IP}$  means).

variations, the locations visited by a customer during the planning period are relocated closer to the home location. Specifically, we take the line segment between a customer’s home location and a location visited by the customer during the planning period and relocate that location on the line segment at distance  $0.5d$ ,  $0.25d$ , and  $0.125d$ , respectively, where  $d$  is the original distance between the two locations. Furthermore, because the travel time between the locations is reduced, the time windows at the locations are increased.

For each of these four variants of an instance, we solve the VRPRDL using HEURRDL and compare the resulting schedule’s cost to the cost of a home delivery (HD) schedule in which the customer remains at home throughout the planning horizon, solved with a simplified version of HEURRDL. We also consider a schedule that allows either home or roaming delivery (HRDL) to the customers, where delivery to the home can occur at any point in the planning horizon, but roaming deliveries must respect the time windows; this corresponds to the delivery company having the option to either deliver to the customer’s car, or to leave the package at home. Table 5 outlines how these costs compare across instance variants. A few table entries are missing; for these instances there was no feasible solution to the VRPRDL. (By relocating the customer locations closer to the home location, an instance can become infeasible.)

We observe that the comparison between pure home and pure roaming delivery (HD/RDL) is mixed, showing that either delivery system can outperform the other depending on the instance. On average, roaming delivery is 4 to 6% more expensive in the extreme cases (farthest and closest customer locations), and 4 to 5% less expensive in the middle cases. We conjecture the following explanations for this behavior. For the instance variants with locations closest to the customer’s home location, the roaming locations are close enough to the home location that traveling to them is tantamount to visiting the home location, but the RDL instance is somewhat restricted by the time windows, while the HD location has no such restriction. Conversely, in the variants with locations farthest away from the customer’s home location, the customer spends a significant amount of the planning horizon traveling between locations, and hence the delivery time windows are narrow and restrict the RDL instance’s scheduling possibilities.

Since the HD and RDL systems exhibit somewhat complementary advantages, it is perhaps not surprising that the combination of the two delivery systems (HRDL) offers significant benefits. The results indicate that this combination can significantly reduce delivery costs over home delivery regardless of whether RDL is more or less expensive, and the cost savings are proportional to how far locations are from the customer’s home. For example, in the variants with locations farthest away, the savings are over 20% on average, even though RDL by itself is more expensive. Figures 4 and 5 show Instance 15 and its HD and HRDL solutions, respectively, in which the flexibility to deliver to the trunk of the car reduced the delivery cost by more than 50%. These results indicate that roaming delivery can significantly impact costs if deployed properly, perhaps in conjunction with home delivery instead of as a replacement.

Next, we focus on the realistic instances, using two different incarnations of each instance. In the original instance, the depot is located in the center of the region (Downtown Atlanta). In the variation, the depot is located in the southern part of the city. By comparing results for the two variants, we can investigate how sensitive these are to the location of the depot. The results can be found in Table 6 where, in addition to the cost ratios HD/RDL and HD/HRDL, we also report the cost ratio RDL/HRDL.

We observe that for the realistic instances the results are quite different than for the general instances. Trunk delivery, whether considered by itself or in combination with home delivery, offers significant cost reductions, more than 65%, on average, when the depot is in the center of the region, and around 40%, on average, when the depot is in the southern part of the city. As one illustration, Figure 6 shows the home delivery and roaming delivery solutions for realistic Instance 7 when the depot is in the center. Unsurprisingly, the cost reductions are smaller

Table 5: Comparison of roaming delivery to home delivery on general instances.

Instance	100% ( $d$ )			50% ( $0.5d$ )			25% ( $0.25d$ )			12.5% ( $0.125d$ )		
	HD RDL	RDL HRDL	HD HRDL	HD RDL	RDL HRDL	HD HRDL	HD RDL	RDL HRDL	HD HRDL	HD RDL	RDL HRDL	HD HRDL
1	0.917	1.159	1.063	1.003	1.066	1.069	1.055	1.000	1.055	0.917	1.127	1.033
2	1.302	1.012	1.317	1.255	1.007	1.264	1.082	1.000	1.082	1.317	0.798	1.052
3	1.187	1.071	1.271	1.041	1.022	1.064	0.921	1.123	1.034	1.186	0.853	1.012
4	0.919	1.214	1.117	1.065	1.026	1.092	1.056	1.000	1.056	0.927	1.116	1.034
5	0.959	1.126	1.079	0.983	1.092	1.074	1.019	1.017	1.037	0.959	1.063	1.019
Group geom. mean	1.046	1.114*	1.165*	1.065	1.042*	1.110*	1.025	1.027	1.053*	1.050	0.981	1.030*
6	0.856	1.446	1.238	1.044	1.056	1.102	1.039	1.024	1.063	0.858	1.198	1.028
7	0.963	1.176	1.133	0.942	1.116	1.051	1.021	0.992	1.013	0.963	1.055	1.017
8	0.929	1.251	1.163	1.092	1.000	1.092	1.051	0.985	1.036	0.981	1.021	1.002
9	0.959	1.272	1.220	1.071	1.037	1.111	1.060	1.002	1.062	0.964	1.074	1.035
10	1.135	1.070	1.215	1.045	1.110	1.160	1.053	1.000	1.053	1.113	0.944	1.051
Group geom. mean	0.964	1.237*	1.193*	1.037	1.063*	1.103*	1.045*	1.001	1.045*	0.972	1.055	1.026*
11	0.996	1.060	1.056	1.039	1.000	1.039	1.021	1.002	1.023	0.996	1.010	1.006
12	0.905	1.422	1.287	0.997	1.233	1.229	1.039	1.008	1.047	0.904	1.129	1.021
13	1.299	1.198	1.556	1.237	1.074	1.328	1.162	1.036	1.205	1.295	0.793	1.027
14	1.002	1.127	1.129	0.992	1.150	1.141	1.073	1.041	1.117	1.001	0.998	0.999
15	1.323	1.205	1.595	1.067	1.090	1.164	1.119	1.019	1.140	1.367	0.861	1.177
16	0.854	1.249	1.066	1.167	1.028	1.199	1.090	1.036	1.129	0.861	1.302	1.121
17	1.013	1.125	1.141	1.087	1.006	1.094	1.032	0.978	1.009	1.018	0.996	1.013
18	0.976	1.239	1.210	1.129	1.062	1.199	1.091	1.012	1.103	0.978	1.056	1.032
19	1.141	1.260	1.438	1.134	0.997	1.130	1.034	1.034	1.069	1.144	0.904	1.034
20	1.119	1.361	1.523	1.185	1.021	1.211	1.028	1.111	1.143	1.120	0.914	1.024
Group geom. mean	1.053	1.220*	1.285*	1.101*	1.064*	1.171*	1.068*	1.027*	1.097*	1.058	0.987	1.044*
21	1.143	1.193	1.364	-	-	1.188	1.038	1.083	1.124	1.151	0.894	1.029
22	0.885	1.297	1.148	1.079	1.103	1.189	1.061	1.041	1.105	0.929	1.119	1.039
23	0.833	1.481	1.234	1.040	1.197	1.245	1.072	1.074	1.151	0.840	1.223	1.028
24	0.796	1.395	1.111	1.078	1.068	1.151	1.038	1.006	1.045	0.812	1.268	1.029
25	0.795	1.613	1.282	-	-	1.253	1.103	1.026	1.132	0.821	1.354	1.112
26	1.003	1.380	1.384	-	-	1.106	1.045	1.068	1.116	1.032	1.058	1.092
27	0.836	1.361	1.138	-	-	1.058	0.993	1.043	1.035	0.837	1.216	1.018
28	0.826	1.343	1.109	0.928	1.237	1.148	1.085	1.067	1.157	0.885	1.249	1.105
29	0.887	1.256	1.114	0.926	1.166	1.080	0.964	1.079	1.040	0.890	1.194	1.063
30	0.935	1.426	1.334	1.000	1.248	1.248	1.040	0.996	1.036	0.971	1.059	1.028
Group geom. mean	0.888*	1.370*	1.217*	1.007	1.168*	1.165*	1.043*	1.048*	1.093*	0.911*	1.156*	1.054*
31	0.864	1.426	1.232	1.150	1.070	1.230	-	-	1.143	0.902	1.204	1.086
32	0.782	1.368	1.070	-	-	1.109	0.938	1.035	0.971	0.832	1.309	1.089
33	0.937	1.251	1.173	-	-	1.140	1.019	1.102	1.123	1.004	1.117	1.121
34	0.829	1.286	1.066	1.034	1.027	1.061	1.037	0.983	1.020	0.861	1.201	1.034
35	0.636	1.681	1.070	0.865	1.237	1.070	0.986	1.025	1.011	0.665	1.542	1.025
36	0.851	1.382	1.177	1.036	1.122	1.162	1.031	1.061	1.094	0.895	1.186	1.061
37	0.953	1.358	1.295	1.001	1.107	1.108	1.015	1.020	1.035	0.999	1.056	1.055
38	0.748	1.404	1.050	0.991	1.105	1.095	1.055	1.032	1.089	0.769	1.387	1.067
39	0.989	1.432	1.416	1.147	1.168	1.339	1.140	1.052	1.199	1.027	1.083	1.112
40	0.825	1.434	1.182	0.990	1.135	1.123	0.986	1.038	1.024	0.852	1.203	1.025
Group geom. mean	0.835*	1.398*	1.168*	1.023	1.120*	1.141*	1.022	1.038*	1.069*	0.874*	1.221*	1.067*
<b>Overall geometric mean</b>	<b>0.94*</b>	<b>1.29*</b>	<b>1.21*</b>	<b>1.05*</b>	<b>1.09*</b>	<b>1.15*</b>	<b>1.04*</b>	<b>1.03*</b>	<b>1.08*</b>	<b>0.96*</b>	<b>1.09*</b>	<b>1.05*</b>

\* significant difference at the 0.05 level (paired t-test using log of ratios).

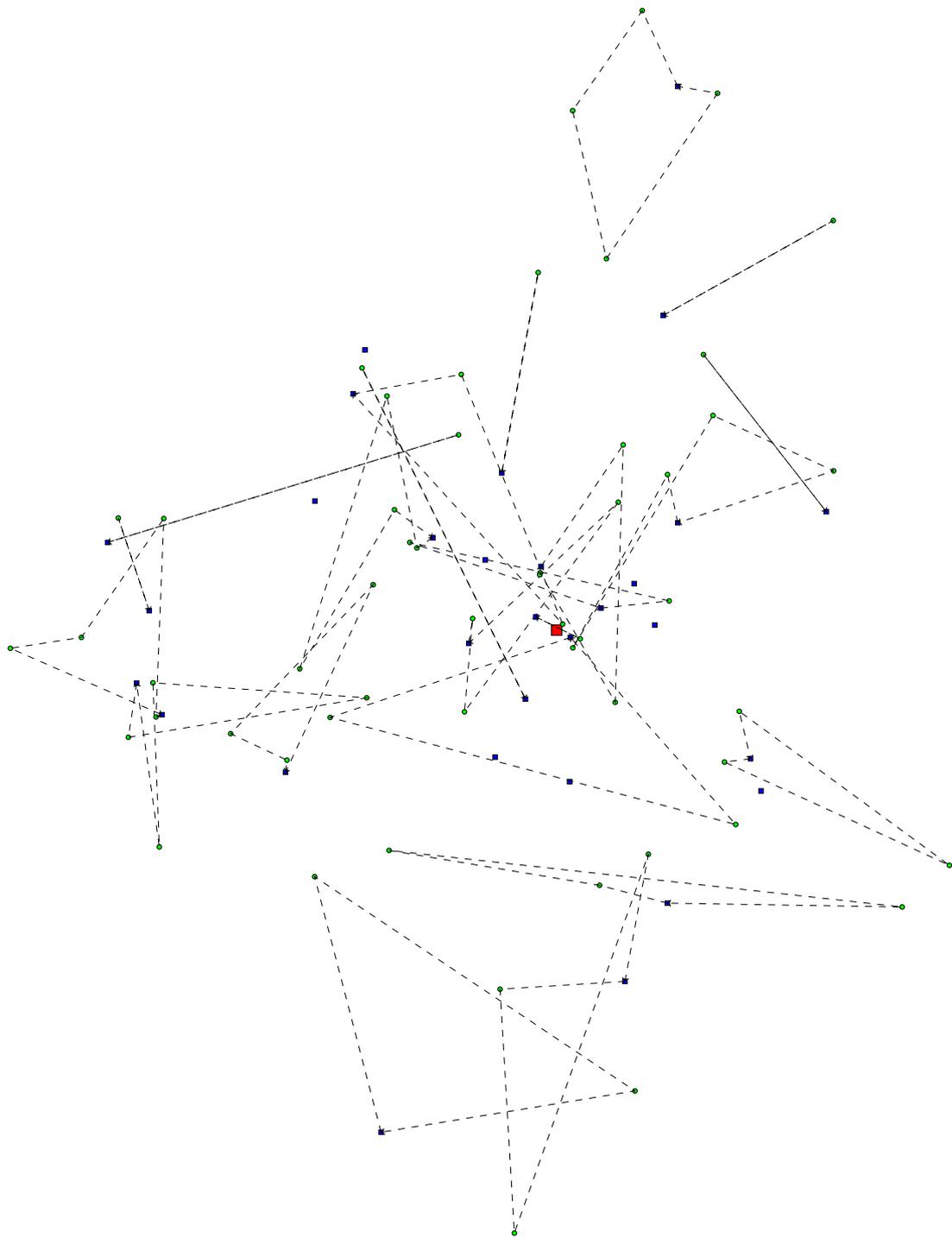
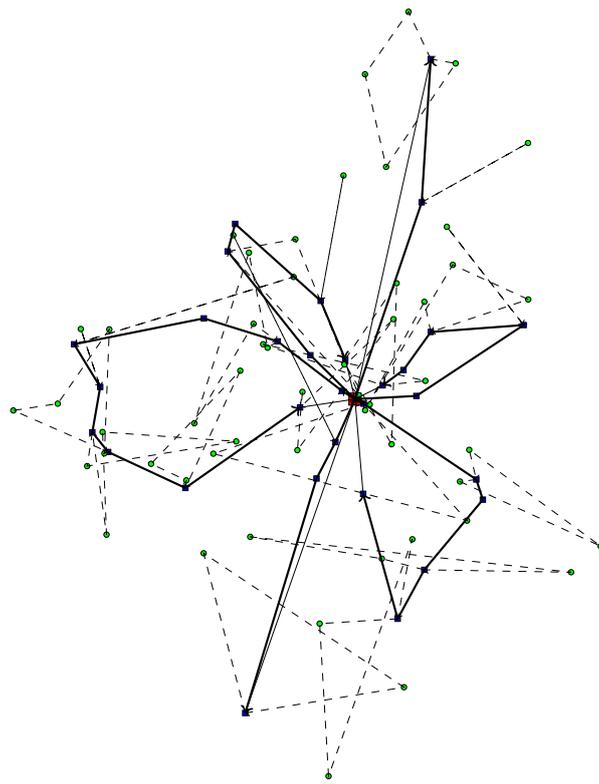
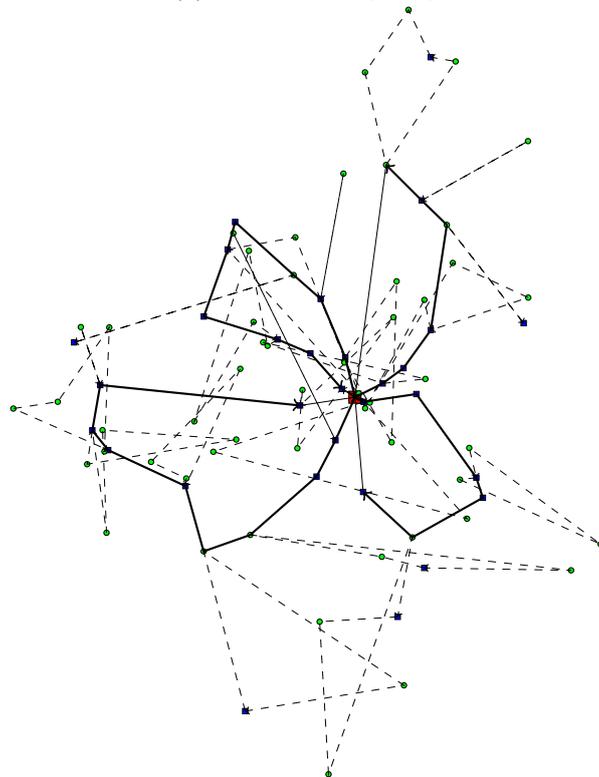


Figure 4: Instance 15 (depot - red square, home location - blue square, roaming location - green circle).



(a) Home delivery only



(b) Home and roaming delivery

Figure 5: HD and HRDL solutions for Instance 15 (routes - thick solid lines, not showing final return to depot).

Table 6: Comparison of roaming delivery to home delivery on realistic instances.

instance	Depot located at (0,0)			Depot located at (0,-30)		
	HD	RDL	HD	HD	RDL	HD
	RDL	HRDL	HRDL	RDL	HRDL	HRDL
r1	1.901	1.021	1.940	1.596	0.991	1.582
r2	1.748	1.074	1.876	1.395	1.145	1.597
r3	1.454	1.099	1.599	1.382	1.037	1.433
r4	1.647	1.02	1.679	1.426	1.044	1.489
r5	1.766	0.976	1.722	1.410	1.018	1.436
r6	1.635	1.109	1.812	1.401	1.050	1.470
r7	1.745	1.131	1.973	1.537	1.009	1.550
r8	1.896	0.926	1.756	1.516	1.028	1.558
r9	1.521	1.018	1.549	1.371	0.985	1.351
r10	1.882	1.004	1.891	1.544	1.024	1.581
Group geom. mean	1.713*	1.036	1.774*	1.456*	1.032*	1.503*
r11	1.380	1.123	1.550	1.348	0.990	1.334
r12	1.711	0.991	1.695	1.396	1.029	1.436
r13	1.774	1.022	1.814	1.395	1.052	1.467
r14	1.858	1.046	1.944	1.519	1.047	1.591
r15	1.651	1.025	1.692	1.344	1.072	1.441
r16	1.587	1.018	1.615	1.361	1.033	1.407
r17	1.579	1.061	1.675	1.458	0.974	1.420
r18	1.779	0.988	1.758	1.470	0.974	1.431
r19	1.651	0.979	1.615	1.339	1.031	1.380
r20	1.832	1.067	1.956	1.517	0.995	1.509
Group geom. mean	1.674*	1.031*	1.727*	1.413*	1.019	1.440*
r21	1.558	0.995	1.550	1.298	1.106	1.435
r22	1.614	1.015	1.639	1.296	1.011	1.310
r23	1.411	0.993	1.402	1.284	1.053	1.352
r24	1.606	1.006	1.616	1.332	0.999	1.331
r25	1.544	0.994	1.535	1.283	1.061	1.361
r26	1.560	1.058	1.651	1.311	1.028	1.348
r27	1.514	1.026	1.553	1.211	1.078	1.305
r28	1.536	1.073	1.648	1.322	1.061	1.403
r29	1.656	0.967	1.601	1.360	1.002	1.362
r30	1.764	0.975	1.720	1.343	0.946	1.270
Group geom. mean	1.574*	1.010	1.589*	1.303*	1.034*	1.347*
r31	1.717	1.016	1.745	1.399	1.007	1.409
r32	1.663	0.992	1.649	1.289	1.095	1.411
r33	1.659	0.974	1.616	1.322	1.018	1.345
r34	1.858	1.038	1.928	1.541	0.947	1.460
r35	1.632	1.046	1.707	1.368	1.041	1.424
r36	1.646	1.010	1.664	1.434	0.973	1.396
r37	1.764	1.008	1.777	1.351	1.108	1.497
r38	1.667	1.004	1.675	1.350	1.019	1.375
r39	1.691	0.985	1.666	1.336	1.082	1.445
r40	1.709	0.997	1.704	1.441	0.985	1.420
Group geom. mean	1.699*	1.007	1.711*	1.381*	1.026	1.418*
<b>Overall geometric mean</b>	1.664*	1.021*	1.699*	1.387*	1.028*	1.426*

\* significant difference at the 0.05 level (paired t-test using log of ratios).

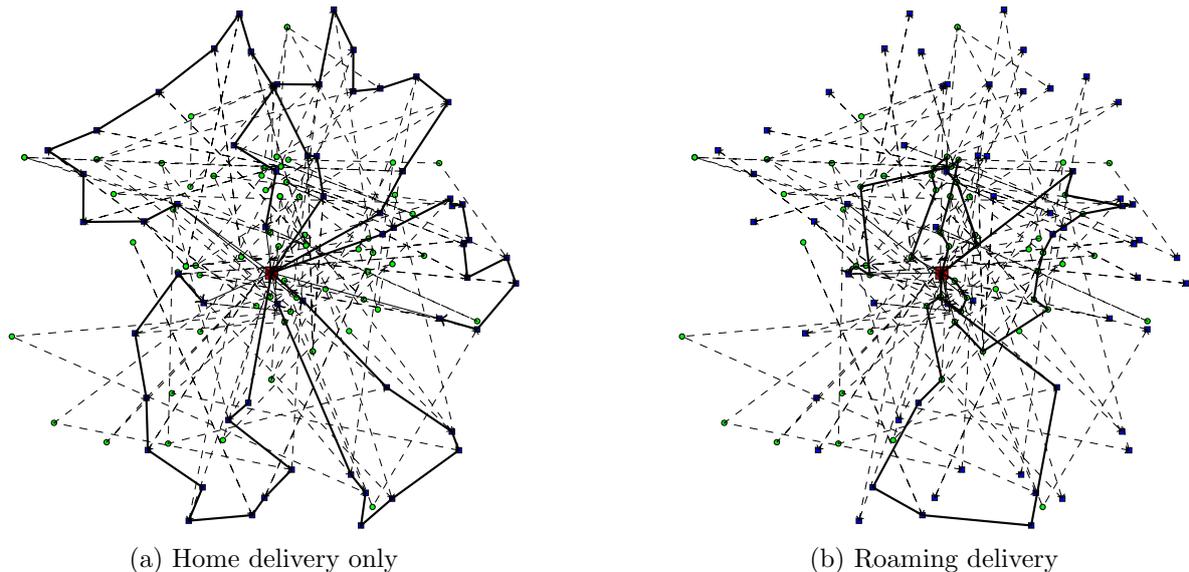


Figure 6: HD and RDL solutions for Realistic Instance 7 (routes - thick solid lines, not showing final return to depot).

with a depot in the southern part of the region than with one in the center of the region: the potential for cost reductions when delivering to customers with a home location in the northern part of the region is, relatively speaking, smaller in the former case.

Interestingly, for these realistic instances, there is no noticeable difference between trunk delivery by itself or trunk delivery in combination with home delivery. This is in stark contrast to what we observed for the general instances, where trunk delivery in combination with home delivery clearly outperformed trunk delivery by itself. This may be explained by the clustering of work locations in the realistic instances, which offer significant opportunities to combine deliveries in very efficient routes; our results thus suggest that RDL may offer the most benefit in cities with a small number of work location clusters but more dispersed residential locations.

## 6. Final remarks

In this paper, we introduced the VRPRDL as a canonical optimization model for the type of routing problem faced by companies deploying a trunk delivery system. The VRPRDL is a special case of the generalized VRP with time windows, in which the locations visited by a customer (more precisely the customer’s car) define the clusters. The fact the time windows of the locations in a cluster reflect the travel itinerary of a customer and are, therefore, non-overlapping, imposes significant (additional) structure. The VRPRDL constitutes a challenging routing problem with important applications in last-mile delivery.

Our computational study suggests that trunk delivery offers a significant opportunity for last-mile package delivery companies to reduce the distance traveled and thus reduce both economic cost and emissions (and likely reduce congestion as well). Even for general instances that do not exhibit realistic geographic patterns, trunk delivery can significantly reduce distance traveled when compared to traditional home delivery. However, the true benefit of trunk delivery is even clearer in instances that reflect more realistic home and work geographical patterns; for these instances, our results suggest that trunk delivery alone could potentially reduce the distance traveled by 40% to 65%, depending on the location of the depot. These reductions in miles traveled would also have a significant environmental and social benefit.

Our results motivate a variety of questions. One example is whether it is possible to precisely characterize when a trunk delivery system is cheaper than a home delivery system and by how much: While our results lend credence to the intuition that trunk delivery can significantly outperform home delivery when many roaming delivery locations are clustered together (as in an office park area), there may be other important factors at play that can be uncovered through further statistical analysis.

In this initial assessment of the benefits of truck delivery, we have assumed that the travel itineraries of customers are known in advance and with complete certainty. Many companies, e.g. Roadie ([www.roadie.com](http://www.roadie.com)), are monitoring the travel behavior of individuals 24 hours a day and are developing machine learning technology to use the collected travel data to predict the daily travel itineraries of these individuals. However, predicted travel itineraries will not be perfect – for instance, an individual may leave work 30 minutes earlier than expected – and this suggests important avenues for future research, including the planning of robust trunk delivery routes and dynamically adjusting trunk delivery routes based on real-time customer travel information.

## References

- Audi, 2015. *Audi, DHL and Amazon deliver convenience*. <https://www.audiusa.com/newsroom/news/press-releases/2015/04/audi-dhl-and-amazon-deliver-convenience>, accessed: 2016-07-24.
- Baldacci, R., Mingozzi, A., Roberti, R., 2012. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research* 218 (1), 1–6.
- Bektaş, T., Erdoğan, G., Røpke, S., 2011. Formulations and Branch-and-Cut Algorithms for the Generalized Vehicle Routing Problem. *Transportation Science* 45, 299–316.
- Biggs, J., Oct 2012. *Cardrops Is A Service That Puts Stuff You Order Into The Trunk Of Your Car. Yeah. Really.* <https://techcrunch.com/2012/10/29/cardrops-is-a-service-that-puts-stuff-you-order-into-the-trunk-of-your-car-yeah-really/>, accessed: 2017-03-30.
- Campbell, A., Savelsbergh, M., 2004. Efficient Insertion Heuristics for Vehicle Routing and Scheduling Problems. *Transportation Science* 38, 369–378.
- Cerrone, C., Cerulli, R., Golden, B., 2014. Carousel Greedy: A Generalized Greedy Algorithm for Optimizing the Cardinality of a Set, Presented at Graphs and Optimization IX Meeting, Sirmione, Italy. [http://scholar.rhsmith.umd.edu/sites/default/files/bgolden/files/carousel\\_greedy.pdf](http://scholar.rhsmith.umd.edu/sites/default/files/bgolden/files/carousel_greedy.pdf).
- Dantzig, G., Ramser, J., 1959. The Truck Dispatching Problem. *Management Science* 6, 80–91.
- Davies, A., Feb 2016. *Volvo Ditches the Car Key to Make Way for the Future*. <https://www.wired.com/2016/02/volvo-kills-the-car-key-to-make-way-for-the-future/>, accessed: 2017-03-30.
- Desrochers, M., Lenstra, J., Savelsbergh, M., Soumis, F., 1988. Vehicle Routing with Time Windows: Optimization and Approximation. In: Golden, B., Assad, A. (Eds.), *Vehicle Routing: Methods and Studies*. Elsevier Science Publishers B.V., North-Holland, pp. 65–84.
- Etherington, D., Sep. 2016. *Daimler begins testing Smart car trunk delivery service with DHL*. <https://techcrunch.com/2016/09/02/daimler-begins-testing-smart-car-trunk-delivery-service-with-dhl>, accessed: 2017-02-27.
- Farber, M., 2016. *Consumers are now doing most of their shopping online*. <http://fortune.com/2016/06/08/online-shopping-increases/>, accessed: 2016-07-24.
- Feo, T., Resende, M., 1995. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization* 6, 109–133.
- Figliozzi, M., 2012. The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E: Logistics and Transportation Review* 48 (3), 616 – 636.  
URL <http://www.sciencedirect.com/science/article/pii/S1366554511001426>
- Gleyo, F., 2015. Volvo in-car delivery will send your shopping goods straight to your car trunk. *Tech Times*.
- Ichoua, S., Gendreau, M., Potvin, J.-Y., 2003. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research* 144 (2), 379 – 396.  
URL <http://www.sciencedirect.com/science/article/pii/S0377221702001479>
- Korosec, K., May 2016. *Volvos Solution for the Package Theft Epidemic: Your Cars Trunk*. <http://fortune.com/2016/05/10/volvo-urb-it-delivery/>, accessed: 2017-02-27.
- Laporte, G., 2009. Fifty Years of Vehicle Routing. *Transportation Science* 43 (4), 408–416.
- Malandraki, C., Daskin, M., 1992. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science* 26, 185–200.

- Moccia, L., Cordeau, J.-F., Laporte, G., 2012. An incremental tabu search heuristic for the generalized vehicle routing problem with time windows. *Journal of the Operational Research Society* 63 (2), 232–244.
- Pisinger, D., Røpke, S., 2007. A General Heuristic for Vehicle Routing Problems. *Computers and Operations Research* 34, 2403–2435.
- Savelsbergh, M., 1986. Local search for routing problems with time windows. *Annals of Operations Research* 4, 285–305.
- Solomon, M. M., 1987. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research* 35 (2), 254–265.
- United Parcel Service of America, Inc., July 2016. UPS 2015 corporate sustainability report. [https://sustainability.ups.com/media/ups-pdf-interactive/UPS\\_2015\\_CSR.pdf](https://sustainability.ups.com/media/ups-pdf-interactive/UPS_2015_CSR.pdf).
- UPS, comScore, June 2016. UPS Pulse of the Online Shopper. [https://solvers.ups.com/assets/2016\\_UPS\\_Pulse\\_of\\_the\\_Online\\_Shopper.pdf](https://solvers.ups.com/assets/2016_UPS_Pulse_of_the_Online_Shopper.pdf).
- Vidal, T., Crainic, T., Gendreau, M., Prins, C., 2013. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers and Operations Research* 40, 475–489.
- Volvo Cars, Feb 2014. volvo cars demonstrates the potential of connected cars with deliveries direct to peoples cars. Press Release, {<https://www.media.volvocars.com/global/en-gb/media/pressreleases/139114/volvo-cars-demonstrates-the-potential-of-connected-cars-with-deliveries-direct-to-peoples-cars>}.
- Wal-Mart Stores, Inc., 2016. Walmart 2016 global responsibility report. <http://corporate.walmart.com/2016grr>.

## A. Instance generation

---

**Algorithm A.1:** GENERATING GENERAL INSTANCES

---

**Input:**

$n$ , number of customers  
 $m$ , maximum number of locations per customer  
 $s$ , constant speed of all vehicles  
 $T$ , planning horizon  
 $p$ , travel distance control ( $0 < p \leq 1$ )  
 $avgD$ , mean demand  
 $stdD$ , standard deviation of demand

**Output:**

VRPRDL instance, as a set of customer profiles with a demand quantity and a sequence of locations and associated time windows (first and last location are home)

**foreach** *customer*  $c \leftarrow 1, \dots, n$  **do**

$d_c \leftarrow$  Random sample from normal( $avgD, stdD$ )

$home_c \leftarrow$  Random point in a circle with radius  $sT/2$  and center at the origin

$m_c \leftarrow$  Random integer in  $\{1, \dots, m\}$

**if**  $m_c = 1$  **then** // customer has a single location

$locsequence_c \leftarrow (home_c)$

$timewindows_c \leftarrow ([0, T])$

**end**

**else** // customer has multiple locations

$dailylocs_c \leftarrow$  Random points in circle of radius  $sTp/2m$  and center at  $home_c$

$locsequence_c \leftarrow (home_c, dailylocs_c, home_c)$

$timewindows_c \leftarrow$  Randomly partition the time available among all locations,  
        where time available is  $T$  minus the total travel time required to visit  $dailylocs_c$

**end**

**end**

---

---

**Algorithm A.2:** GENERATING REALISTIC INSTANCES

---

**Input:**

$n, n_2, n_3$ , number of customers, and percent of customers with 2 and 3 locations  
 $W_1$ , percent of customers working 4 hours and work start  $\in \{8 \text{ am}, \text{noon}\}$   
 $W_2$ , percent of customers working 4-7 hours and work start  $\in \text{Uniform}[8 \text{ am}, 10 \text{ am}]$   
 $WC$ , set of work cluster center coordinates  
 $K_2, S_2$ , number and minimum separation of locations in each work cluster  
 $K_3, S_3$ , number and minimum separation of after-work locations  
 $p_1, p_2, p_3$ , maximum proportion of time a customer can travel 1) before work, 2) after work (total), 3) after work (direct to home distance)  
 $T, T_1, T_2, T_3$ , planning horizon and radii of homes, work, and after-work clusters  
 $avgD, stdD$ , mean and standard deviation of demand

**Output:** VRPRDL instance, as a set of customer profiles with a demand quantity and a sequence of locations and associated time windows (first and last location are home)

**foreach**  $c \leftarrow 1, \dots, n$  **do**

$m_c \leftarrow$  number of locations that customer  $c$  will have, according to  $n, n_2, n_3$

**end**

**foreach**  $c \leftarrow \text{RandomOrder}(1, \dots, n)$  **do**

$ws_c, we_c \leftarrow$  start and end of work time window, according to  $W_1, W_2$ .

**end**

$L_1$  : initialize set of potential home locations

$L_2$  : initialize set of potential work locations

$L_3$  : initialize set of potential after-work locations

$L_1 \leftarrow$  points in circular area of radius  $T_1$  centered at origin

**foreach**  $wc$  in  $WC$  **do**

$L_2 \leftarrow L_2 \cup \{K_2 \text{ points within } T_2 \text{ radius about } wc \text{ and with } S_2 \text{ minimum separation} \}$

**end**

$L_3 \leftarrow \{K_3 \text{ points within } T_3 \text{ radius from origin and with } S_3 \text{ minimum separation} \}$

**foreach**  $c \leftarrow 1, \dots, n$  **do**

$d_c \leftarrow$  Random sample from normal( $avgD, stdD$ )

**if**  $m_c = 1$  **then** // customer has a single location

$home_c \leftarrow$  uniform-random sample point from  $L_1$

$locsequence_c \leftarrow (home_c)$

$timewindows_c \leftarrow ([0, T])$

**end**

**else if**  $m_c = 2$  **then** // customer has 2 locations

$w_c \leftarrow$  random sample from  $L_2$

$h_c \leftarrow$  uniform-random sample point from  $L_1$  and close enough to  $w_c$ , given  $p_1, p_2$

$locsequence_c \leftarrow (h_c, w_c, h_c)$

$timewindows_c \leftarrow ([0, ws_c - tt(h_c, w_c)], [ws_c, we_c], [we_c + tt(w_c, h_c), T])$

**end**

**else** // customer has 3 locations

$w_c \leftarrow$  random sample from  $L_2$

$h_c \leftarrow$  uniform-random sample point from  $L_1$  and close enough to  $w_c$ , given  $p_1, p_2$

$a_c \leftarrow$  random sample from  $L_3$  close enough to  $w_c$  and  $h_c$ , given  $p_2, p_3$ .

$as_c = we_c + tt(w_c, a_c)$ .  $x \leftarrow$  uniform-random in

$[0, (T - we_c - tt(w_c, a_c) - tt(a_c, h_c))]$

$locsequence_c \leftarrow (h_c, w_c, a_c, h_c)$

$timewindows_c \leftarrow$

$([0, ws_c - tt(h_c, w_c)], [ws_c, we_c], [as_c - x, as_c + x], [as_c + x + tt(a_c, h_c), T])$

**end**

**end**

---

Table B.7: Number of successes for the improvement neighborhoods employed by HEURRDL.

Instance	RDGR	Switch (DP)	rGDrGR
1	8	3	1
2	7	0	0
3	3	0	5
4	4	0	1
5	6	0	0
6	3	0	0
7	9	1	2
8	6	0	8
9	3	0	0
10	2	0	1
11	12	2	2
12	23	0	1
13	13	1	1
14	15	4	1
15	15	1	0
16	14	1	1
17	18	2	2
18	20	2	1
19	25	0	3
20	25	2	1
21	55	5	0
22	60	7	0
23	36	1	0
24	34	2	0
25	31	0	0
26	34	6	0
27	37	3	0
28	46	3	0
29	37	0	0
30	35	1	0
31	64	3	0
32	89	0	0
33	64	2	0
34	83	1	0
35	72	0	0
36	92	2	0
37	86	3	0
38	72	1	0
39	110	2	0
40	69	1	0

## B. Algorithm performance details

Here, we delve deeper into the experiment results for the general instances and provide statistics on the contribution of the two problem-specific algorithmic ideas incorporated in the construction and improvement algorithms, i.e., employing the dynamic program to optimize the locations visited for a given sequence of customers during a switch of neighborhoods, and maintaining a number of alternative schedules to be considered during insertion and deletion operations. In Table B.7, for the two neighborhoods employed during the improvement phase of HEURRDL we present the number of times they were active when a schedule improvement was found as well as the number of times the dynamic program improved the schedule when switching from one neighborhood to the other.

In Table B.8, we present the fraction of insertion and deletion operations in which the flexibility to switch to an alternative location was exploited. We further distinguish between whether the switch to another location was exploited for the customer preceding or succeeding the customer being inserted or deleted.

Table B.8: Percentage of multi-location switches.

Instance	% alt. pre. ins	% alt. pre. del	% alt. suc. ins	% alt. suc. del
1	0	0	9	9
2	0	6	13	13
3	0	0	0	0
4	5	0	23	29
5	3	0	0	31
6	4	1	1	0
7	19	20	2	2
8	17	19	11	13
9	18	18	10	10
10	6	6	3	0
11	6	6	21	21
12	10	10	22	24
13	7	6	2	3
14	2	4	10	11
15	5	3	2	4
16	9	10	19	20
17	2	9	9	9
18	2	3	14	25
19	10	12	5	6
20	4	4	6	7
21	2	2	2	2
22	1	1	6	9
23	2	3	2	2
24	9	8	4	4
25	1	2	7	9
26	3	4	7	7
27	10	9	5	6
28	3	2	11	19
29	4	6	4	5
30	6	7	1	1
31	2	3	4	5
32	2	3	2	2
33	2	2	3	4
34	3	4	1	1
35	3	3	3	5
36	2	2	2	3
37	1	1	5	5
38	2	3	1	1
39	3	3	2	3
40	4	4	3	2