Chapter 8

How to assess your Smart Delivery System?

Benchmarks for rich vehicle routing problems

Luis A.A. Meira, Paulo S. Martins, Mauro Menzori, and Guilherme A. Zeni

School of Technology (FT), University of Campinas (UNICAMP), Limeira, SP, Brazil

8.1 Introduction

Benchmarks are found in various fields of science, such as reservoir engineering [15,21], economics [30], and climatology [45], among other areas. They play a central role in computer science, for example, in image processing [18, 35,27], hardware performance [9], and optimization [41,34,7].

Within the context of optimization, Johnson [29] divided algorithm analysis into three approaches: the worst-case, the average-case, and the experimental analysis. Regarding experimental papers, he identifies four cases: (i) solving a real problem; (ii) providing evidence that one algorithm is superior to others; (iii) better understanding a problem; and (iv) studying the average case. He proposes the use of well-established benchmarks to provide evidence of the superiority of an algorithm (item ii). Such papers are called horse-race papers.

Johnson highlights that reproducibility and comparability are essential aspects of any experimental paper. The author mentions the difficulty in justifying experiments on problems with no direct application. Such problems have no real instances, and the researcher is forced to generate the data in a vacuum.

Barr et al. [5] indicate several purposes in an optimization research. The proposed algorithm may be fast, accurate, robust, simple, high-impact, generalizable, or innovative. They reinforce that results on standard benchmark problems and problems of practical interest should always be a part of an algorithmic experiment.

Our work deals with a variant of the Vehicle Routing Problem (VRP) based on a real mail delivery case in the city of Artur Nogueira. The post office receives thousands of mail items to be delivered on a daily basis. Such a mail is distributed to a set of 15–25 mail carriers for on-foot delivery. Each mail carrier is modeled as a vehicle, and each delivery point is a customer. The problem
is named here as Post Office Deliveries VRP (PostVRP). It is a multiobjective variant of the distance-constrained VRP.

Domain experts indicate that the PostVRP has three main objective functions to be minimized (while maintaining the feasibility of the solutions): (i) route length; (ii) unfairness, measured as the workload (i.e., route length) variance among the mail carriers; and (iii) number of mail carriers.

The PostVRP considers uncapacitated vehicles and constrained route length. Each mail carrier is allowed to carry a maximum load from 8 to 10 kg. A support truck restocks the mail carriers turning their capacities unlimited. Each mail carrier must follow a 6-to-8-hour working day, which implies a maximum capacity for the route length.

A limited route length is a constraint that models several real-world cases: A helicopter has a route length limited by the capacity of its fuel tank. Workers, in general, have a time window to operate the vehicle, which likewise limits the length of the route.

This work presents a Smart Delivery System (SDS) case study modeled in a Brazilian city located at $22°34'22''S$,$47°10'22''W$. The proposed benchmark contains up to 30,000 customers. We make available the benchmark tool so that it is possible to create new arbitrarily large instances. The methodology can be applied to other cities and to other VRP variants.

The remainder of this chapter is organized as follows: the background and review of relevant work is provided in Section 8.2; in Section 8.3, we introduce the notation and definitions; Section 8.4 presents the model; and Section 8.5 addresses a real-world PostVRP benchmark case. Finally, we summarize and conclude in Section 8.6.

8.2 Literature review

One of the first references to the VRP dates back to 1959 [17], under the name Truck Dispatching Problem, a generalization of the Traveler Salesman Problem (TSP). The term VRP was first seen in the paper by Christofides [10]. Christofides defines VRP as a generic name, given to a class of problems that involves the visit of “customers” using vehicles.

Real-world aspects may impose variants to the problem. For example, the Capacitated-VRP (CVRP) considers a limit to the vehicle capacity [22], the VRP with Time Windows (VRPTW) accounts for the delivery time windows [32], and the Multi-Depot VRP (MDVRP) extends the number of depots [43].

Jozefowiez et al. [31] surveys the existing research related to multiobjective optimization in routing problems. Amaya et al. [1] introduces the capacitated arc routing problem with refill points. The vehicle servicing arcs must be refilled on the spot by using a second vehicle. Gusmmag-Pfiegl et al. [25] create heuristics for a real-world mail delivery problem. Four distinct vehicles are considered: car, moped, bicycle, and walking. Other VRP variants may be easily found in the literature.
Reinelt [41] created a benchmark for the TSP known as TSPLib. In his work he consolidated unsolved instances from 20 distinct papers. His repository, named TSPLIB95 [42], has instances of both symmetric and asymmetric Traveling Salesman Problems (TSP and aTSP) and three related problems: (i) CVRP; (ii) Sequential Ordered Problem (SOP); and (iii) Hamiltonian Cycle Problem (HCP).

The number of instances is 113, 19, 16, 41, 9 for TSP, aTSP, CVRP, SOP, and HCP, respectively. The number of vertices varies from 14 to 85,900 for the TSP, from 17 to 443 for the aTSP, from 7 to 262 for the CVRP, from 7 to 378 for the SOP, and from 1000 to 5000 for the HCP.

The optimum of all TSPLib instances was finally achieved in 2006, after fifteen years of notable progress in algorithm development. The optimum of the d15112 instance was found in 2001 [3]. This instance contains 15,112 German cities, and it required 22.6 years of processing split across 110 500 MHz processors [14]. The pla33810 instance was solved in March 2004 [3]. It represents a printed circuit board with 33,810 nodes (Fig. 8.1), and it was solved in 15.7 years of processing [19]. The last instance of the TSPLib, called pla85900, was solved in 2006 [3]. This instance contains 85,900 nodes representing a VLSI application.

Solomon [44] created a benchmark for the VRPTW in 1987. It is composed of 56 instances partitioned into six sets: R1, C1, RC1, R2, C2, and RC2. The data are randomly generated in problem sets R1 and R2, cluster-generated in problem sets C1 and C2, and generated as a mix of random and cluster structures in problem sets RC1 and RC2. The number of customers is 100 in all instances. The vehicle has a fixed capacity, and the customers have an integer demand. The number of vehicles is not fixed: it is derived from the fact that capacity is limited. Under this viewpoint, this can be considered a multiobjective problem. It aims to minimize the route and the number of vehicles.

The first optimum solution was published by Kohl et al. [33]. Chabrier [8] solved 17 open instances in the benchmark. Amini et al. [2] obtained solutions very close to the optimum, considering only the first 25 customers. In July 2015, 28 years after having launched the benchmark, Jawarneh and Abdullah [28] published a Bee Colony Optimization metaheuristic. Such algorithm reached 11 new results in extended Solomon’s VRPTW instances. It is surprising that such small instances present a quite complex internal structure to be optimized. Fig. 8.2 shows a Solomon’s instance composed of 100 customers and a given solution considering three vehicles.

Regardless of their complexity, the TSPLib and Solomon benchmarks have a number of customers between 100 and 262 for the VRP, which is currently a small value. Gehring and Homberger [23] extended the Solomon’s instances, thus creating a benchmark for the VRPTW with the number of customers varying from 100 to 1000.
Nguyen et al. [39] proposed a genetic algorithm to optimize the World TSP challenge. It is a TSP instance with 1,904,711 locations throughout the world. Each location is specified by its latitude and longitude.
For the CVRP, ABEFMP is a largely used set of instances, in which Augerat et al. [4] proposed the A, B, P classes in 1995, and Christofides and Eilon [12], Fisher [20], Christofides [11] proposed the E, F, and M classes in 1969, 1994, and 1979, respectively. In their benchmark the number of customers varies from 13 to 200, and the number of vehicles varies from 2 to 17.

Fukasawa et al. [22] and Contardo and Martinelli [13], among others, obtained the optimum in different ABEFMP instances. Pecin et al. [40] found the optimum solution for the last unsolved instance, named M-n151-k12, 35 years after its presentation by Christofides [11]. Despite that, most of those instances are very simple to solve nowadays.

Golden et al. [24] proposed new instances for the CVRP. It is a set of 20 instances, with the number of customers varying from 240 to 483. Such a benchmark remains entertaining, because most of its instances still do not have an optimum established [46]. Li et al. [37] created a set of instances with the number of customers between 560 and 1200. Currently, no optimum has been found for any of the instances [46].

Uchoa et al. [46] created the CVRPLib, where they consolidated the CVRP instances of [4,12,11,20,24,37]. In addition, Uchoa et al. [47] generated new instances with the number of customers between 100 and 1000. Their work indicates the lack of well-established challenging benchmarks for the VRP.

Uchoa et al. also highlight the fact that benchmarks are artificially created. Solomon and Uchoa et al. generated their own instances using random points. In the ABEFMP benchmark some random instances are generated, and other instances represent real problems. However, in all the instances the customers are points in the Euclidean space. The instances of Golden et al. [24] and Li et al. [37] are artificial as well.

Cwiek et al. [16] created a heuristic benchmark generator for VRPTW. Each delivery is \((x_i, y_i) \in \mathbb{R}^2\), and the distance between two deliveries is the Manhattan metric \(c_{ij} = |x_i - x_j| + |y_i - y_j|\). Kytöjoki et al. [36] found solutions for artificial VRP instances with up to 20,000 customers and real-world instances with up to 12,000 customers. This is probably the largest VRP instance reported in the literature.

The SINTEF website contains best known results for standard benchmarks of several variants of the Vehicle Routing Problem (https://www.sintef.no/projectweb/top/).

### 8.3 Notation and definition

Consider a set of elements \(S\) where a depot is a special element \(\pi \in S\). This work does not address multiple-depot variants to the VRP. The set of customers is defined by \(C = S \setminus \{\pi\}\), and the number of customers is denoted by \(n\), where \(C = \{c_1, \ldots, c_n\}\). The number of vehicles in the fleet is represented by \(k \in \mathbb{N}\). The value \(k\) is traditionally considered a constant, but it is possible to define variants to VRP where \(k\) is variable. Let \(w : S \times S \rightarrow \mathbb{N}\) be the cost between
any two elements in \( S \). Let \( S(C, k) = (c_1, \ldots, c_n, \pi, \ldots, \pi) \). This sequence is created as follows: (i) all elements in \( C \) are inserted in \( S \); (ii) the depot vertex is inserted \( k - 1 \) times.

Each permutation of \( S(C, k) \) represents a solution to the VRP. For example, consider the graph and vertices described in Fig. 8.3, and suppose that the number of vehicles is three (i.e., \( k = 3 \)). The \( S(C, 3) \) sequence is given by \( S(C, 3) = (c_1, \ldots, c_{13}, \pi, \pi) \). For example, the permutation \( S' = (c_3, c_5, c_4, c_1, c_2, \pi, c_6, c_{10}, c_{11}, c_{12}, \pi, c_7, c_8, c_9, c_{13}) \) is the solution described in Fig. 8.3.

All routes begin and end at the depot. The \( S' \) solution represents a partition of the clients in three routes: \( R_1 = (c_3, c_5, c_4, c_1, c_2) \), \( R_2 = (c_6, c_{10}, c_{11}, c_{12}) \), and \( R_3 = (c_7, c_8, c_9, c_{13}) \). The vertex \( \pi \) is used to create a partition of the sequence in \( k' \leq k \) routes. Let \( \text{Partition}(S) = (R_1, \ldots, R_{k'}) \). By definition empty routes are not part of \( \text{Partition}(S) \). Thus, \( \text{Partition}(1, 2, \pi, \pi, 3, 4) \) is \( \{(1, 2), (3, 4)\} \) and not \( \{(1, 2), (1, 3, 4)\} \), that is, \( k' \leq k \).

The length of a route \( R = (r_1, \ldots, r_m) \) is given by

\[
W(R) = w(\pi, r_1) + w(r_m, \pi) + \sum_{i=1}^{m-1} w(r_i, r_{i+1}).
\]

The length of a solution \( S = (s_1, \ldots, s_m) \) is the travel distance and is calculated as

\[
W(S) = \sum_{R \in \text{Partition}(S)} W(R).
\]

The number of vehicles used in a given solution is equal to the number of nonempty routes \( |\text{Partition}(S)| \). If the number of vehicles is \( k \) and nonempty routes are allowed, then we have the constraint \( |\text{Partition}(S)| = k \). If the number of vehicles are at most \( k \), or if empty routes are allowed, then we have \( |\text{Partition}(S)| \leq k \). If the number of vehicles is not a part of the input, then the
How to assess your Smart Delivery System?

Chapter 8

233

domain may be given by the permutation of the sequence \( S(C, n) \). In this case the number of vehicles is defined during the optimization step.

Given a feasible solution, it is necessary to calculate its costs. The most common objective function to be minimized is the length of the solution:

\[ f_1(S) = W(S). \]

Another objective function consists in finding a feasible solution that minimizes the number of vehicles:

\[ f_2(S) = |\text{Partition}(S)|. \]

Suppose there are 25 available mail carriers and a feasible solution with 21 routes. In this case the post office may allocate the four available mail carriers to other internal tasks.

Finally, it is required that the solution meet the fairness criteria, that is, routes should be assigned in a way that balances out the workload (route length) among the mail carriers. The way we modeled fairness was through minimizing the variance of the route lengths:

\[ f_3(S) = \sqrt{\frac{\sum_{R \in \text{Partition}(S)} (W(R) - \bar{W})^2}{|\text{Partition}(S)| - 1}}. \]

VRP is a set of problems that consists of visiting customers using vehicles. Each variant has additional feasibility constraints, such as not allowing empty routes (\(|\text{Partition}(S)| = k\)). The PostVRP assumes that the length of the route is limited. Let \( R_{\text{max}} \) be the maximum allowed route length, that is, \( W(R) \leq R_{\text{max}} \).

**Definition 1** (PostVRP). Given a set of elements \( S \), a weight function \( w : S \times S \to \mathbb{N} \), a constant \( k \in \mathbb{N} \) representing the maximum number of vehicles, a special vertex \( \pi \in S \), and a maximum route length \( R_{\text{max}} \in \mathbb{N} \). Let \( C \leftarrow S \setminus \{\pi\} \). Consider the sequence \( S(C, k) \), and let \( P_e \) be the set of all feasible permutations of \( S(C, k) \) with respect to \( R_{\text{max}} \). The PostVRP problem consists of minimizing \((f_1(S), f_2(S), f_3(S))\) subject to \( S \in P_e\).

### 8.4 Model description

This section describes the PostVRP model. We start the process of creating the benchmark by mapping each street onto a street map graph. Each street \( S_f \) is modeled as a polygonal chain, which is defined as a set of planar coordinates. For example, suppose a University St. modeled as a polygonal chain \( P = (c_1, \ldots, c_n) \), where \( c \in \mathbb{R}^2 \) for all \( c \in P \). The complete map graph has a set \( \mathcal{P} = (P_1, \ldots, P_n) \) of polygonal chains, one for each street.
We create a graph \( G(V, E) \) based on \( P \). Each vertex \( v \in V \) is associated with a Cartesian coordinate \( (x_v, y_v) \in \mathbb{R}^2 \), and each edge \( e = (u, v) \) is a straight line segment between \( u \) and \( v \). The edge weight is \( w'(e) = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2} \). The vertices associated with corners are automatically built by a line segment intersection algorithm.

Fig. 8.4 (right) displays simple streets modeled as one polygonal chain and streets with islands, which are modeled by two parallel polygonal chains. Additional segments are added to allow shortcuts in footpaths.

**FIGURE 8.4** A section of the city map (left). Edges and vertices created over the city map (right).

Given an edge \( e \), its street is denoted by \( St(e) \). Each street \( St \) has an arbitrarily defined width \( \text{wth}(St) \in \mathbb{R}^+ \), which represents the cost of crossing the street. The value \( \text{wth}(St) \) can be set to zero, thus resulting in no cost to cross the street.

A nonnormalized probability density \( D(St) \) is assigned to each street. The probability of a street receiving a delivery workload per unit length is directly proportional to the density value \( D \). Such a value is used to create a central street with a large workload compared to a distant one. The probabilities are outlined in the next subsection.

### 8.4.1 Generating delivery points

Consider a street map graph \( G(V, E) \). The probability of one delivery being assigned to an edge \( e \), denoted by \( \text{Prob}(e) \), is

\[
\text{Prob}(e) = \frac{D(St(e))w'(e)}{T}, \text{ where } T = \sum_{e' \in E} D(St(e'))w'(e').
\]

\( \text{Prob}(e) \) is directly proportional to the edge length \( w'(e) \) and to the probability density \( D(St(e)) \), and it must be normalized to obtain \( \sum_{e \in E} \text{Prob}(e) = 1 \).

The location of a given delivery \( d \), denoted by \( \text{loc}(d) \), is composed of three attributes: an edge \( (u, v) \), a value \( \alpha \in [0, 1] \), and a label \( \text{street} \_\text{side} \in \{\oplus, \ominus\} \). The delivery is positioned at the affine combination of \( u \) and \( v \) in respect to \( \alpha \), that is, \( (x_u, y_u)(\alpha) + (1 - \alpha)(x_v, y_v) \). The street of a delivery \( d = (e, \alpha, \text{street} \_\text{side}) \), denoted by \( St(d) \), is the street of the edge \( St(e) \).
An integer $n$ represents the number of deliveries, and an artificial delivery $d_π$ is created for the depot. The value of $α$ is randomly generated within the interval $[0, 1]$. The street side label is an equiprobable random choice in the set $\{⊕, ⊖\}$. Algorithm 1 is then used to create the delivery set. Given that the number of deliveries is a part of the input, it is possible to set arbitrarily large instances.

**Input:** An integer $n$ and a set of edges $E$ with probabilities $\text{Prob}(e)$, $\forall e \in E$

**Output:** A set of deliveries $\text{Del}$.

1. $\text{Del} \leftarrow \emptyset$
2. Partition all edge probabilities in the interval $[0, 1]$ for $i = 1$ to $n$
3. Select a random value $r \in [0, 1]$
4. If $r$ is in the interval associated with $\text{Prob}(e)$ then
5. Select a random value $α \in [0, 1]$
6. Select a random street side value $s \in \{⊕, ⊖\}$
7. $\text{Del} \leftarrow \text{Del} \cup \{(e, α, s)\}$
8. end
9. end
10. return $\text{Del}$

Algorithm 1: Algorithm to create deliveries.

### 8.4.2 Defining the weight between a pair of deliveries

The street map graph $G(V, E)$ is used to compute the weight between deliveries. Given two deliveries $d_a$ and $d_b$, the cost to cross the street is defined as

$$\text{cross}(d_a, d_b) = \begin{cases} wth(St(d_a)) & \text{if } (d_a, d_b) \text{ side labels are } \{(⊕, ⊖), (⊖, ⊕)\}, \\ 0 & \text{and } St(d_a) = St(d_b), \\ \end{cases}$$

A constant $β \in \mathbb{R}^+$, which represents an additional fixed cost per delivery, must be defined. The weight between two deliveries $d_a = (e_a, α_a, s_a)$ and $d_b = (e_b, α_b, s_b)$ is given by $w(d_a, d_b)$. If $e_a = e_b$, then

$$w(d_a, d_b) = |α_a - α_b|w'(e_a) + \text{cross}(d_a, d_b) + β. \quad (8.1)$$

Let $G(V, E)$ be the original street map, $e_a = [u_a, v_a]$, $e_b = [u_b, v_b]$, and let $G^*(V^*, E^*)$ be defined as $V^* = V \cup [d_a, d_b]$, $E^* = E \cup \{(u_a, d_a), (v_a, d_a), (u_b, d_b), (v_b, d_b)\}$ (Fig. 8.5); thus

$$w(d_a, d_b) = \minpath(d_a, d_b, G^*) + \text{cross}(d_a, d_b) + β. \quad (8.2)$$

The instance is composed of a matrix $w_{n \times n}$, an integer $k$, and an integer $R_{max}$. The first delivery represents the depot.
8.4.3 The benchmark tool

This subsection describes the tool that creates the benchmark. It has three configuration files, `background.png`, `model.txt`, and `instances.txt`. The background file contains an image used to improve visualization, and its resolution is used as the base for the model. The model file must contain the following information:

- Depot location: the coordinate position reference to the depot;
- Additional cost per delivery: cost to hand out the delivery;
- Decimal precision: number of digits after the fractional part;
- Pixel value: value used to convert a pixel into other units. This value may represent the speed of the vehicle;
- Attributes: attributes used to compute the street probability density and the cost to cross the street;
- Roadmap: the description of streets including the polygonal chain.

A researcher may perform experiments where pixel value, additional cost per delivery or decimal precision are variable. Consider a white background with 500 × 500 pixels and the model described in Fig. 8.6. The tool will process the model file and create a roadmap (Fig. 8.7). The depot is positioned at the closest edge. The probability density $D$ of the 4th Av is 0.4 because it is a [AV, PERIPHERAL, RADIOACTIVE] with values $\{20, 0.2, 0.1\}$.

The last file is named `instance.txt` (Table 8.1). Each line corresponds to an instance in the benchmark. It must contain the instance ID, the directory and subdirectory, the maximum number of vehicles, and a comment line. Each line must also contain a pseudo-random generator seed and an MD5 signature. The MD5 checksum value is used to ensure the instance identity. The tool will recreate the instances offline and verify the MD5 signature in the instance file. The tool will execute the files of Fig. 8.6 and Table 8.1 and create the instances shown in Fig. 8.8.

We can edit the instance file to create new instances for a given model. Once the new instances are created, it is necessary to manually update the MD5 signature. For instance, a new seed will create a new instance with another pseudo-random sequence.
How to assess your Smart Delivery System?

Chapter 8

FIGURE 8.6 Example of a model file.

```plaintext
#GENERAL
DEFINE DEPOT_X 250
DEFINE DEPOT_Y 250
DEFINE ADDITIONAL_COST_PER_DELIVERY 10
DEFINE DECIMAL_PRECISION 6
#ONE PIXEL IS PIXEL_VALUE_UNITS (meters/sec, etc).
DEFINE PIXEL_VALUE 10.0
#PENALTIES
DEFINE PERIPHERAL .2
DEFINE RADIOACTIVE .1
#STR
DEFINE STR 10
DEFINE STR_CROSSCOST 20
#AVE
DEFINE AVE 20
DEFINE AVE_CROSSCOST 10

#ROADMAP
1th STR [STR][100,100]-[499,100]
2th STR [STR,PERIPHERAL][100,250]-[499,250]
3th STR [STR,RADIOACTIVE][100,300]-[499,300]
4th STR [STR,PERIPHERAL,RADIOACTIVE][100,450]-[499,450]
1th AVE [AVE][100,100]-[499,100]
2th AVE [AVE,PERIPHERAL][200,100]-[499,100]
3th AVE [AVE,RADIOACTIVE][300,100]-[499,100]
4th AVE [AVE,PERIPHERAL,RADIOACTIVE][400,100]-[499,100]
```

FIGURE 8.7 Map based on the model in Fig. 8.6.

TABLE 8.1 Instance file.

<table>
<thead>
<tr>
<th>ID</th>
<th>Dir</th>
<th>Subdir</th>
<th>n</th>
<th>k</th>
<th>Rmax</th>
<th>Comment</th>
<th>Seed</th>
<th>MD5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ex</td>
<td>ex_0_0</td>
<td>0</td>
<td>0</td>
<td>2941.15</td>
<td>Max route [...]</td>
<td>100</td>
<td>4d...af</td>
</tr>
<tr>
<td>1</td>
<td>ex</td>
<td>ex_10_5</td>
<td>10</td>
<td>5</td>
<td>2941.15</td>
<td>The size [...]</td>
<td>101</td>
<td>e9...10</td>
</tr>
<tr>
<td>2</td>
<td>ex</td>
<td>ex_100_5</td>
<td>100</td>
<td>5</td>
<td>2941.15</td>
<td>Consider [...]</td>
<td>102</td>
<td>eb...a2</td>
</tr>
<tr>
<td>3</td>
<td>ex</td>
<td>ex_1000_5</td>
<td>1000</td>
<td>5</td>
<td>2941.15</td>
<td>Instance [...]</td>
<td>103</td>
<td>05...62</td>
</tr>
<tr>
<td>4</td>
<td>ex</td>
<td>ex_10000_5</td>
<td>10000</td>
<td>5</td>
<td>2941.15</td>
<td>Instance [...]</td>
<td>104</td>
<td>93...53</td>
</tr>
</tbody>
</table>

The project site [38] provides the source program that parses the instance file. The program executes a simple swap optimization and saves the route in
both text and image files (Fig. 8.9). The purpose is to provide the researcher with a parse to the instance file and a visualization of the solution.

8.4.4 Modeling Manhattan (NY) streets

This section shows how the methodology can be adapted to other benchmark cases through a pilot sample that indicates its possible uses. We start off with an image of a section of Manhattan (Fig. 8.10).

The streets (34 in this case) are manually added as a polygonal chain in the model file. As the streets are generally straight, they are composed mostly by two pixel coordinates. We label the streets with four types with respective penalties and cost to cross the streets (Fig. 8.11).

Fig. 8.12 contains one instance with 1000 deliveries based on this model.
FIGURE 8.10  Section of the original map.

FIGURE 8.11  Manhattan model file.

8.5 Real-world PostVRP benchmark (RWPostVRPB)

In this section the tool is used to model a real-world mail delivery in the city of Artur Nogueira, Brazil, namely RWPostVRPB. To make the instances as real-
istic as possible, the authors relied upon domain expertise from an actual post office in the aforementioned city.

We build the model from a PDF image instead of using existing street map graphs for the following reasons (Fig. 8.13): (i) the path on foot may differ from those available from graphs which prioritize delivery by vehicles; (ii) the number of streets in Artur Nogueira is sufficiently small to allow the manual creation of the graph (≈ 400 streets); and (iii) currently, public maps such as OpenStreetMap [26] are incomplete, that is, the city has a large number of streets not covered so far.

The tool automatically computes corners resulting in a graph with $|V| = 2111$ and $|E| = 3225$. Each vertex is associated with a pixel, and each edge is associated with a straight line between the two edge pixels. The cost of an edge $(u, v)$ is directly proportional to the Euclidean distance between vertices $u$ and $v$ in $\mathbb{R}^2$.

Each street is classified using the Region (R), Type (T), and Zone (Z) attributes. Each attribute has a corresponding number of levels, and each attribute-level pair is associated with a multiplicative penalty (Pen) in $\mathbb{R}^+$. Table 8.2 contains the assignment of attribute, level, and penalty based on expert knowledge.

In the proposed model the streets located in the downtown area have a higher delivery rate per unit length than those located in the outskirts. Such behavior is captured by the Region attribute through four levels: central, peripheral, distant, and isolated. The Type attribute has also four levels, namely avenue, street, way, and highway, whereas the Zone attribute may be commercial, mixed, and residential. We used Google Maps as an auxiliary tool to classify streets. Each of the 422 streets received a value in $R \times T \times Z$ according to expert knowledge.
The (nonnormalized) probability density $D : Streets \to \mathbb{R}^+$ is obtained from the multiplicative penalties. For example, the 15th Avenue is \textit{(central,}...
TABLE 8.2 Attribute, level and penalty values.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Level 1 (Pen)</th>
<th>Level 2 (Pen)</th>
<th>Level 3 (Pen)</th>
<th>Level 4 (Pen)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region (R)</td>
<td>central (1.0)</td>
<td>periphery (75)</td>
<td>distant (4)</td>
<td>isolated (2)</td>
</tr>
<tr>
<td>Type (T)</td>
<td>large Avn (1.0)</td>
<td>Avenue (75)</td>
<td>Street (4)</td>
<td>Drive (0)</td>
</tr>
<tr>
<td>Zone (Z)</td>
<td>commercial (1.0)</td>
<td>mixed (75)</td>
<td>residential (4)</td>
<td>–</td>
</tr>
</tbody>
</table>

large Avn, mixed), and thus \( D(15th) = 1 \times 1 \times 0.7 = 0.7 \). On the other hand, Jasmine street is (isolated, Street, residential), and thus \( D(Jasmine) = 0.2 \times 0.2 \times 0.4 = 0.16 \). In this example a random delivery to the 15th avenue is more probable than a delivery to Jasmine way by unit length.

The RWPostVRPB contains 78 instances divided into four groups: Toy, Normal, OnStrike, and Christmas (Table 8.3). The Toy set contains 30 instances with a small number of deliveries, and it may be used to validate algorithms before their use with realistic and larger instances. The maximum number of vehicles is 30, 45, and 60 for Normal, OnStrike, and Christmas, respectively.

TABLE 8.3 RWPostVRPB instances description.

<table>
<thead>
<tr>
<th>Set</th>
<th># Instances</th>
<th># Deliveries</th>
<th>Length (hrs)</th>
<th># Vehicles (max)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toy</td>
<td>30</td>
<td>3 to 5000</td>
<td>6</td>
<td>5 to 15</td>
</tr>
<tr>
<td>Normal</td>
<td>15</td>
<td>10,000 to 14,000</td>
<td>6</td>
<td>30</td>
</tr>
<tr>
<td>OnStrike</td>
<td>15</td>
<td>15,000 to 19,000</td>
<td>8</td>
<td>45</td>
</tr>
<tr>
<td>Christmas</td>
<td>18</td>
<td>20,000 to 30,000</td>
<td>8</td>
<td>60</td>
</tr>
</tbody>
</table>

The Normal set contains 15 instances from 10,000 to 14,000 deliveries. Artur Nogueira city has around 50,000 inhabitants and an average of 12,000 daily mail deliveries. The normal daily work of a mail carrier has eight hours a day. The mail carrier spends two hours preparing the deliveries inside the post office and six hours to complete the deliveries on foot. The post office has around 15 mail carriers to perform the deliveries. The number of vehicles is variable in PostVRP with a defined maximum value. If a feasible solution with 11 mail carriers is found, then the post office may assign four mail carriers \((15 - 11)\) to internal tasks.

The OnStrike and Christmas sets may be used to model contingencies. The OnStrike set is similar to Normal, but the number of deliveries is larger (from 15,000 to 19,000), and the maximum route length is eight hours. The Christmas set models special seasons with high delivery rates. The post office often hires extra mail carriers for the Christmas season. A feasible solution with 17 mail carriers represents two new hires \((15 + 2)\). For all sets, a minimum average route length is desired as well as a minimum variance between the lengths of the routes. Fig. 8.14 shows the distribution of 10,000 delivery points for a chosen area of Artur Nogueira.
The speed of the vehicles were set as 0.69 s/pixel, that is, \( \text{PIXEL\_VALUE}=0.69 \). The fixed cost per delivery was defined as \( 5 \times 0.69 \text{ s} = 3.45 \text{ s} \), that is, \( \text{ADDITIONAL\_COST\_PER\_DELIVERY}=5 \).

The full set of instances can be downloaded from the project website [38]. The website also includes a pilot sample modeling a section of Manhattan NY.

On July 10, 2017, each instance was available in the website as a matrix \( w \), where \( w(u, v) \) is the cost to make a delivery in \( v \) starting from \( u \) for all \( u, v \in S \). For the largest instance, the matrix is \( 30,000 \times 30,000 \) using 10.4 GB of memory. On January 17, 2018, we made available the instance represented as a roadmap graph \( G(V, E, w) \). Each delivery is represented as a triple \( (e, a, \text{street\_side}) \), and the distance between two deliveries is computed by Eqs. (8.1) and (8.2). The first representation has a higher level of disorder than the second one. A third representation is the configuration files and the benchmark tool.

### 8.6 Final remarks and conclusion

In this chapter we have worked on the PostVRP, a multiobjective VRP variant with a route length constraint. The three objectives to be minimized were (i) the number of vehicles, (ii) the solution length, and (iii) the standard deviation of the length of the routes.
Informally, we list some desirable characteristics of a good benchmark: simple, generalizable, incremental complexity, well-defined objective function, it does not have private information that reduces search space and must have a model and a support tool.

A “simple benchmark” has a reasonable amount of information without excess. Thus we opted to model the map in the 2D plan instead of a 3D model. We could also have assumed that speed slows down with human fatigue, among other possibilities. Too much details can be negative.

The “generalization” concept refers to the application of the methodology to other situations. The proposed methodology is suitable for PostVRP, but it can be adapted to other variants of the VRP. The map is created as a set of polygonal chains, so it is relatively simple to model other cities. It is also possible, with an additional effort, to work with directed edges.

Incremental complexity is an advantage, since it is possible to analyze the limits of an algorithm. An exact algorithm can optimize relatively small instances compared to heuristic methods. A heuristic algorithm $O(n^3)$ must optimize smaller instances than another of complexity $O(n^2)$. An incremental complexity benchmark allows analyzing the boundaries of different algorithms.

For a well-defined objective function, we use rounding to work only with integer values. Even the variance can be converted to integer values. Minimizing $\sigma$ is equivalent to minimizing $\sigma^2n(n-1)$, with the second option having integer image. In addition, a java algorithm was provided that parses the input file and calculates the value of the objective functions.

To compare two algorithms, it is necessary that they receive the same information about the benchmark. If an algorithm has a hint that allows a warm start, then the results are not comparable. In this work we offer three representations of the instances. The first consists of a matrix $w_{n \times n}$, and the second, much more compact, has the roadmap graph and deliveries. An algorithm that uses the graph has a competitive advantage over an algorithm that works only with the $w$ matrix. In this way we may say that the problems are different according to the available information.

In this research we proposed a benchmark where a model and a tool are both available. The tool provides a toy optimization algorithm with the calculation of the objective function and visualization mechanisms for the instances. This way of representing the instances can be generalized to other problems. The variables in the file instances.txt contain parameters of the instances such as number of deliveries, maximum length of the route, and a seed.

By using the tool we created a benchmark that models a problem that comprised the mail delivery on foot in a Brazilian city. The instances were classified into four groups: (i) Toy, with up to 5000 deliveries, (ii) Normal, with up to 14,000 deliveries, (iii) OnStrike, with up to 19,000 deliveries, and (iv) Christmas, with up to 30,000 instances. The benchmark can be used both for comparison and validation of optimization algorithms for routing problems.
The application of the tool allows the generation of arbitrary and large instances in the proposed benchmark by changing the number of deliveries in the instance file. Likewise, new instances can be created by changing the seed of the instance. Additionally, researchers may also model new scenarios.

Acknowledgments

This research was partially supported by FAPESP (Grant Number 2015/11937-9).

References


How to assess your Smart Delivery System?


