VRP in urban areas to optimize costs while mitigating environmental impact

Carmine Cerrone1, Anna Sciomachen

Accepted: 19 April 2022 / Published online: 22 July 2022 © The Author(s) 2022

Abstract
Nowadays, the need to think about sustainable mobility, both goods and people, is widely recognized. For this reason, many recent papers have moved in this direction. In this context, particular attention is now devoted to urban mobility, mainly from a smart city perspective. The present work focuses on sustainable urban freight distribution and proposes a variant of the VRP, which presents some innovative aspects. The goal is to minimize the routes’ cost components, including traveling and external costs due to environmental issues, depending on the chosen vehicles and the different urban streets to cross. In addition, restrictions on the maximum duration of each route to ensure frequent sanitation of vehicles used for deliveries, as required from the beginning of the COVID-19 pandemic, are imposed. The distribution network is modeled by a weighted digraph for which some properties are proved. To face the problem, we present a mixed-integer linear programming model, a math-heuristic associated with it, and a memetic algorithm approach. The results of the reported computational experimentation with random instances specifically tailored for the problem show the efficiency of the proposed methods. Further, test cases based on data of the distribution network of two B2C companies operating in the city of Genoa, Italy, proved the effective application of the proposed methods in the direction of sustainable urban distribution plans.

Keywords Green vehicle routing · Urban distribution · Meta-heuristic

1 Introduction
From the last few years, almost daily, media reports of damage caused by weather and atmospheric phenomena. Public opinion agrees on the need to change their lifestyles towards more environmentally sustainable behavior, while both scientists and politicians call for a reduction of pollutants to protect the environment. Above all, logistics and transport enterprises cannot ignore the need to establish shipping and routing plans aimed at reducing, among others, emissions, noise, and accidents along their paths. In this direction, recent literature has presented many works, having as one of the main goals the minimization of the mentioned above external costs and their evaluation. More precisely, some papers in the last few years proposed either different ways to calculate and evaluate externalities (see, e.g., Bektaş et al. 2019; Bigazzi and Figliozzi 2013; Dekker et al. 2012; Petro and Konečný 2019) or pricing policies to reduce them (see, among others, Kellner et al. 2017; Chang et al. 2018). Other research works presented strategies to incentivize vehicles or transport modalities with lower environmental impact (Okushima 2018; Xu et al. 2019). In particular, taking into account also the social need and the impact that distribution of goods has on citizens, in the current decade, many studies have been proposed aimed at sustainable urban mobility, especially from a smart city perspective (Carrabs et al. 2017; Kauf 2019; Szymczyk and Kadlubek 2019; Strale 2019).

It is true that smart mobility is a crucial topic able to impact the well-being of citizens and the environmental policies of the cities (Taniguchi et al. 2012). As part of the policies implemented in the context of smart mobility to reduce the environmental impact, the distribution of goods, and more precisely the home delivery due to e-commerce, has focused a lot of attention. The goal is twofold: to analyze the likely impact of distribution operations in urban areas upon pollu-
tion and to see how sustainable freight distribution can be assumed (Figliozi 2010; Cerulli et al. 2018; Heshmati et al. 2019).

From an operations research and management science point of view, the Vehicle Routing Problem (VRP) is widely used in decision-making processes about the distribution of goods and services. Today, models and solution methods of VRP are encountered very frequently in the optimization of distribution plans where sustainable factors have to be taken into proper account (Bektas and Laporte 2011; Behnke and Kirschstein 2017; Demir et al. 2014; Ehmke et al. 2018; Erdogan and Miller-Hooks 2012; Koyuncu and Yavuz 2019; Kramer et al. 2015; Yu et al. 2019). In the very recent literature, VRPs have been proposed in which, in addition to the need to plan the distribution of goods in urban areas in a sustainable manner, they also consider the need to contain the spread of the COVID-19 virus (Chen et al. 2020; Cerrone et al. 2021). In accordance with this topic, the present work focuses on sustainable urban freight distribution within a smart city framework and proposes a variant of the Green VRP (GVRP), in which mobility models are suggested by the needs arising from the COVID-19 pandemic are also taken into account. In particular, in the proposed GVRP, we have to choose from an available fleet of non-homogeneous vehicles, those vehicles to deliver goods of a large-scale retailer company in an urban area. Vehicles differ for their size, capacity, and pollution class. In addition, government traffic restrictions and street limitations are given. The goal is to minimize the cost components, which include traveling, loading, and external costs, due to environmental issues, depending on the type of chosen vehicles and urban streets to cross. The present work originates from and extends the results presented in Carrabs et al. (2017), where the authors proposed a Mixed Integer Linear Programming (MILP) model for defining delivery plans in a Business to Consumer (B2C) context. More precisely, we present a refinement of the model given in Carrabs et al. (2017), by including valid inequalities and examining additional constraints, depending on the vehicle type and EURO-Class. The minimization of emissions and pollution reduction is the main objective or part of the generalized cost function in many VRPs. In literature, these problems have often been called Pollution Routing Problem (PRP): Erdogan and Miller-Hooks (2012), Xiao et al. (2012). In this paper, we propose two different approaches derived from the recent literature to solve our GVRP separating two of the most crucial aspect of any optimization technique: the improving phase and the feasibility restoring.

The organization of the paper is as follows. In Sect. 2 we introduce in detail the underlying optimization problem together with the required notation. Some properties concerning the derived network model are also given. In Sect. 3, we give the formulation of the related MILP model. In Sect. 4 the proposed heuristics are given. Section 5 presents the computational experimentations, based on both random instances and real size instances derived from a retail company located in Genoa, Italy. Finally, we give some conclusions and outline future work.

2 Problem definition

As already mentioned, in the present work, we are involved with the definition of sustainable distribution plans of goods in urban areas. This problem can be considered as a variant of the so-called GVRP. More precisely, given a number of customers located in a metropolitan area requiring a known amount of goods and a fleet of non-homogeneous vehicles, we have to determine i) a set of routes originating and terminating at a depot and servicing all the customers and ii) which vehicles to select for traveling along the routes. Vehicles differ for their size, capacity, cost, and pollution class. The goal is to minimize the total cost given by traveling, loading, and pollutant cost components. Note that this goal aims at favoring the use of the most eco-friendly vehicles, especially in the city center and close to schools and parks (Carrabs et al. 2014).

To model the problem we are focused on, a digraph $G = (\mathcal{N}, A)$ defines the underlying urban transportation network, where $\mathcal{N} = \{d\} \cup U$ is the set of $n + 1$ nodes, representing the set $U$ of customers and the depot node $d$; $A$ is the set of $m$ paths of the city connecting pair of nodes.

For each customer $i \in U$, we indicate with $q_i$ the required amount of units of goods. A value $l_{ij}$ is associated with each arc $(i, j) \in A$, representing the length of the corresponding road connections.

Set $V$ contains the fleet of vehicles. To each vehicle, $v \in V$ has associated a quintuple of parameters $(w, e, k, c, s)$ specifying, respectively, its class of weight, pollution emission type, capacity, cost/km, and setup cost. All vehicles are grouped into classes $\mathcal{s}$, such that given two vehicles $v_i, v_j \in V$, $t_{v_i} = t_{v_j}$ if and only if $w_{v_i} = w_{v_j}, e_{v_i} = e_{v_j}, k_{v_i} = k_{v_j}, c_{v_i} = c_{v_j}, s_{v_i} = s_{v_j}$. Note that the class of weight of the vehicles determines the subset of urban paths that are inhibited to them, while the type refers to their category in terms of polluting emissions. Finally, note that the cost of each vehicle depends on both its class of weight and the pollution emission type, which refers to the standard EURO-classes from III to VI (see Sect. 2.1).

The solution to the problem is the set of minimum-cost vehicle routes for delivering the required goods to all customers. As in the classical VRP, some conditions have to be satisfied: (i) all nodes, except node $d$, must be served only once by a vehicle; (ii) all vehicles return to node $d$ after completing their service; (iii) all vehicles have a maximum capacity $k_v$.
It should be noted that in this work, the time windows for the delivery of goods are given implicitly. In fact, the delivery plans are defined considering only the time slots of a day in which in the city of reference is allowed the transit of commercial vehicles. In particular, the time slots available for the delivery of goods vary from 2 to 4 h depending on the vehicles. Furthermore, during this period of the COVID-19 pandemic, the presence of these constraints allows the sanitation of the vehicle (and drivers) several times during the day.

2.1 Cost of arcs

In this work, depending on the pollution emission type \( e \) of each vehicle \( v \), the air pollution \( a_v \) is quantified economically, and its impact is added to the cost of each arc in the graph, also considering the marginal climate change costs \( m_v \). We calculate the costs of the arcs, according to the results shown in Cerulli et al. (2018). In particular, the air pollution and marginal climate change costs are considered with respect to three different cases, corresponding to EURO classes III, IV, and V–VI of the vehicles, respectively. The costs due to air pollution and marginal climate change of the fleet of vehicles expressed in Euro/km are reported in Table 1, and refer to the two main types of commercial vehicles, namely light commercial vehicles (LCV) and heavy good vehicle (HGV). The costs reported in Table 1 have been derived from (Ricardo 2014).

In this scenario the cost of crossing arc \((i, j) \in A\) by vehicle \(v\) is calculated as:

\[
c_{ijv} = (c_v + a_v + m_v)l_{ij}\delta_{ij}.
\]

The parameter \(\delta_{ij} \geq 1\) is associated with each arc \((i, j) \in A\) and is used to increase the traversing cost of a street that passes too close to schools, parks, or other places of interest.

In many real-world scenarios, crossing a road by vehicles with a specific class of weight \(w\) could be prohibited (see Fig. 1). Considering that a cost \(c_{ijv}\) is associated with each vehicle \(v \in V\), in case the road is prohibited for \(v\), logically we set \(c_{ijv} = +\infty\) (see Fig. 1). Obviously in all the procedures implemented in this work, arcs with cost \(c_{ijv} = +\infty\) are not created.

### Table 1

| Environmental cost per Euro/Km of the available vehicles | LCV | HGV-1 | Marginal climate change cost \(m_v\) | LCV | HGV-1 |
|---------------------------------------------------------|-----|-------|-------------------------------------|-----|-------|
| EURO III                                               | 0.046 | 0.061 | 0.028 | 0.029 |
| EURO IV                                                | 0.032 | 0.038 | 0.028 | 0.027 |
| EURO V–VI                                              | 0.011 | 0.017 | 0.028 | 0.025 |

### Algorithm 1 Graph Reduction

1. Input: street network graph \(G = (N, A)\).
2. Create the empty graph: \(\hat{G} = (N, A)\).
3. For each customer \(n \in \hat{N}\) create a corresponding node \(n \in N\).
4. For the depot \(d \in \hat{N}\), create the depot \(d \in N\).
5. For all \(i, j \in N\) do
6. \(A = A \cup \{(i, j), (j, i)\}\)
7. For all \(v \in V\) do
8. \(\hat{SP}_{i,j,v} = \text{Shortest path from } i \text{ to } j \text{ in } \hat{G} \text{ using the costs defined by Eq. (1)}\).
9. \(\hat{SP}_{j,i,v} = \text{Shortest path from } j \text{ to } i \text{ in } \hat{G} \text{ using the costs defined by Eq. (1)}\).
10. \(c_{ijv} = \sum_{(\hat{n}_1, \hat{n}_2) \in \hat{SP}_{i,j}} c_{\hat{n}_1,\hat{n}_2}v\)
11. \(l_{ij} = \sum_{(\hat{n}_1, \hat{n}_2) \in \hat{SP}_{i,j}} l_{\hat{n}_1,\hat{n}_2}\)
12. \(c_{ijv} = \sum_{(\hat{n}_1, \hat{n}_2) \in \hat{SP}_{j,i}} c_{\hat{n}_1,\hat{n}_2}v\)
13. \(l_{ij} = \sum_{(\hat{n}_1, \hat{n}_2) \in \hat{SP}_{j,i}} l_{\hat{n}_1,\hat{n}_2}\)
14. end for
15. end for
16. Return \(G\).

2.2 Graph reduction

The graph \(G = (N, A)\) represents the road network within which our vehicles move. In Sect. 2 we have specified that the set \(N\) of nodes is equal to \(|d| \cup U\). This implies that excluding the depot \(d\), all the nodes of the graph represent customers, and there are no transit nodes.

To transform a generic street network \(\hat{G} = (\hat{N}, \hat{A})\) in graph \(G\), we use the following procedure (see Algorithm 1 - Graph Reduction).

In lines 4 and 5 of the algorithm, according to the customers and the depot of \(\hat{G}\), the corresponding nodes are created in \(G\). At line 6, for each pair of nodes \(i, j \in N\), the arcs \((i, j)\) and \((j, i)\) are created in \(A\). On lines 8 and 9 the shortest paths between nodes \(i, j \in \hat{N}\) related to nodes \(i, j \in N\) are computed. On lines 10 and 12, the cost of each arc \((i, j) \in A\) is created in relation to the specific vehicle \(v\).

Finally, lines 11 and 13 show the procedure for calculating the route length of the paths between each pair of nodes in \(G\) on the basis of the average speed of the vehicles and the service time at the nodes, as described above.

Figure 2 reports an example of how the graph reduction algorithm works. The nodes of the street network \(\hat{G}\) (Fig. 2a) are the depot \(d\), the customers \(\hat{n}_2\) and \(\hat{n}_3\), and the transit nodes \(\hat{n}_1\) and \(\hat{n}_4\). The dashed arc \((\hat{n}_1, \hat{n}_3)\) represents a street prohibited to the transit of HGV vehicles. The weights on the arcs of...
Fig. 1 Example of calculation of the arcs’ cost for two vehicles \(v_1\) and \(v_2\). Dashed lines represent streets that cannot be traveled by HGV vehicles. Therefore, arc \((1, 2)\) is not compatible with the weight class \(w_{v_2}\). \(\delta_{2d}\) is set to 2 since arc \((2, d)\) is adjacent to a school.

\[v_1 = (w_{v_1} = 3.5t, e_{v_1} = LCV\ EURO\ III, k_{v_1} = 50, c_{v_1} = 0.142\€, s_{v_1} = 60\€)\]

\[v_2 = (w_{v_2} = 7.5t, e_{v_2} = HGV\ EURO\ IV, k_{v_2} = 120, c_{v_2} = 0.246\€, s_{v_2} = 90\€)\]

\[c_{d1v_1} = 0.432\]

\[c_{d2v_1} = 0.864\]

\[c_{d1v_2} = 0.622\]

\[c_{d2v_2} = 1.244\]

\[\delta_{12} = 1\]

\[\delta_{d1} = 1\]

\[\delta_{2d} = 2\]

\[l_{d1} = 2\text{Km}\]

\[l_{d2} = 2\text{Km}\]

\[l_{12} = 1\text{Km}\]

**Fig. 2** An example of the Graph Reduction Algorithm. In \(G\) the cost \(c_{dnv_1}\) of crossing arc \((d, n_2)\) \(\in A\) with vehicle \(v_1\), represents in \(\hat{G}\) the cost of the shortest path \((\hat{d}, \hat{n}_1)(\hat{n}_1, \hat{n}_2)\) using the same vehicle \(v_1\). The cost \(c_{dnv_2}\) of crossing arc \((d, n_2)\) \(\in A\) with vehicle \(v_2\), represents in \(\hat{G}\) the cost of the shortest path \((\hat{d}, \hat{n}_1)(\hat{n}_1, \hat{n}_3)(\hat{n}_3, \hat{n}_4)(\hat{n}_4, \hat{n}_2)\) using vehicle \(v_2\).

\(\hat{G}\) are those derived in Fig. 1. Figure 2b depicts the resulting graph \(G\), which is our referring network model. Arcs of \(G\) are the solution of the shortest path problem between customers and customers and the depot, according to lines 8 and 9 of the Graph Reduction algorithm. Note that the cost of the path from the depot to customer \(n_2\) traveled by vehicle \(v_2\) is much higher than the same path traveled by vehicle \(v_1\), due to the traffic limitation of graph \(\hat{G}\).  

**2.3 Graph reduction property**

The graph reduction function allows us to process smaller inputs speeding up the solution approaches. This section will analyze the impact of the transformation on the optimal solution produced by an exact approach.

**Proposition 1** Let \(\hat{G}\) be the graph of the road network and \(G = \text{Graph Reduction}(\hat{G})\) the reduced graph obtained by applying the algorithm (Graph Reduction). The value of the optimal solution calculated on graph \(G\) is greater than or equal to the value of the optimal solution calculated on graph \(\hat{G}\).
VRP in urban areas to optimize costs while mitigating environmental impact

Fig. 3 a Street network graph; nodes $n_1$ and $n_2$ are customers. Maximum length imposed for each route equal to 6. $S = (d, n_1, n_1, n_2)$ optimal solution in $\hat{G}$, cost = 4, length = 6. b Graph Reduction Algorithm example from graph $\hat{G}$ to graph $G$. $S = (d, n_2, n_1)$ optimal solution in $G$, cost = 6, length = 6. c Graph Reduction L1 Algorithm example from graph $\hat{G}$ to graph $G'$. $S = (d, n_1, n_2)$ optimal solution in $G$. Proof Starting from a solution $S$ in $G$ it is always possible to create a solution $\hat{S}$ in $\hat{G}$ with the same objective function value. The assumption is trivial since each route $R \in S$ can be transformed into a route $\hat{R} \in \hat{S}$ by replacing the arcs of $R$ with the corresponding shortest paths in $\hat{G}$, thus keeping the cost of the route, its travel time and its total demand unchanged. This assures that it is not possible to create a solution in $G$ better than the optimal solution presented in $\hat{G}$. To prove this, we will use the example reported in Fig. 3b.

Proposition 2 Let $\hat{G}$ be the graph of the road network and $G = \text{GraphReduction}(\hat{G})$ the reduced graph obtained by applying the algorithm (Graph Reduction). If the cost and the length of each arc are such that:

$$c_{ijv} \leq c_{hkv} \Rightarrow l_{ij} \leq l_{hk} \ \forall (i, j), (h, k) \in \hat{A}, \forall v \in V$$

(2)

then the cost of the optimal solution calculated on graph $G$ is equal to the cost of the optimal solution calculated on graph $\hat{G}$.

Proof Given $\hat{S}$ a feasible solution of our problem on graph $\hat{G}$, let $\hat{R}$ be a sequence of nodes representing a generic route belonging to $\hat{S}$ associated with a generic vehicle $v \in V$. Removing from $\hat{R}$ all nodes with null demand, we will get a new sequence of nodes $R$. Considering that by construction, between each pair of successive nodes in $R$ there is a path in $\hat{G}$, so $R$ is a route also in $G$. Given two consecutive nodes $(n_i, n_j)$ in $R$, in $\hat{R}$ there is the path composed of nodes $(n_i, \ldots, n_j)$. Further, since each arc of $G$ represents the minimum cost path between the same pair of nodes in $\hat{G}$, then the cost $c_{ijv}$ of the arc $(n_i, n_j)$ traveled with vehicle $v \in V$ is less than or equal to the cost of the route from node $n_i$ to node $n_j$ in $\hat{G}$ with the same vehicle. This means that the cost of the route $R$ will always be less than or equal to the cost of the route $\hat{R}$. Based on inequality (2) we can say that the travel time of the arc $(n_i, n_j)$ is less than or equal to the travel time of the path from node $n_i$ to node $n_j$ in $\hat{G}$.

This means that, according to the given travel time constraint, the length (travel time) of route $R$ will be less than or equal to that of route $\hat{R}$. Since routes $\hat{R}$ and $R$ have the same customer demand, the maximum capacity of the vehicle $v \in V$ is never exceeded. With this simple conversion we have shown that it is possible to transform each solution $\hat{S}$ in
\( \hat{G} \) into a solution \( S \) in \( G \) having a value less or equal to the objective function.

Now using proposition (2.3) our proof is complete. \( \square \)

**Algorithm 2** Graph Reduction L1

1: Input: street network graph \( G = (N, A) \).
2: \( \hat{G} = \text{GraphReduction}(G) \).
3: for all \( i, j \in N \) do
4: for all \( v \in V \) do
5: \( \hat{SP}_{i,j,v} = \text{Shortest path from } i \text{ to } j \text{ in } \hat{G} \) with respect to the length of the arcs.
6: \( \hat{SP}_{j,i,v} = \text{Shortest path from } j \text{ to } i \) in \( \hat{G} \) with respect to the length of the arcs.
7: \( l_{ij} = \sum_{(\hat{a}, \hat{a}_2) \in \hat{SP}_{i,j}} l_{\hat{a}\hat{a}_2} \)
8: \( l_{ji} = \sum_{(\hat{a}, \hat{a}_2) \in \hat{SP}_{j,i}} l_{\hat{a}\hat{a}_2} \)
9: end for
10: end for
11: Return \( \hat{G} \).

Starting from the \textit{GraphReduction} algorithm, we created the \textit{GraphReductionL1} algorithm by changing the calculation of the length of the arcs in \( G \). In particular \( \forall (i, j) \in A, \forall v \in V, l_{ij} = \text{minimum path between } i \text{ and } j \text{ in } \hat{G} \) with respect to the length of the arcs in \( \hat{A} \).

**Proposition 3** Let \( \hat{G} \) be the street network graph and \( G = \text{GraphReductionL1}(\hat{G}) \) the reduced graph obtained by applying the algorithm (Graph Reduction L1). Then the cost of the optimal solution calculated on graph \( G \) is less or equal to the cost of the optimal solution calculated on graph \( \hat{G} \).

\[ MIN \sum_{v \in V} c_{ij}x_{ij} + \sum_{v \in V} s_vx_{dv} \quad (3) \]

\[ \text{s.t.} \]
\[ \sum_{i \in N} x_{ijv} = 1 \forall \ j \in U \quad (4) \]
\[ \sum_{i \in N} x_{ijv} - \sum_{j \in N} x_{jiv} = 0 \forall \ k \in U, \forall \ v \in V \quad (5) \]
\[ \sum_{i \in N} y_{ikv} - \sum_{j \in N} y_{jkv} = q_k \sum_{h \in N} x_{hkv} \forall \ k \in U, \forall \ v \in V \quad (6) \]
\[ y_{ijv} \leq (k_v - q_i)x_{ijv} \forall (i, j) \in A, \forall \ v \in V \quad (7) \]
\[ y_{ijv} \geq q_ix_{ijv} \forall (i, j) \in A, \forall \ v \in V \quad (8) \]
\[ \sum_{(i,j) \in A} l_{ijv}x_{ijv} + \sum_{(i,j) \in A; j \in U} L_cv_{ijv} \leq L_c \forall \ v \in V \quad (9) \]
\[ x_{ijv} \in \{0, 1\} \forall (i, j) \in A, \forall \ v \in V \quad (10) \]

In order to model the distribution problem in urban areas defined in the previous session, we define the following decisional variables.

\[ x_{ijv} = \begin{cases} 1 & \text{if vehicle } v \text{ travels from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases} \]

\[ y_{ijv} \in [0, k_v], \text{ representing the amount of goods carried by vehicle } v \text{ from } i \text{ to } j. \]

Let us now give the proposed MILP model of the problem that is derived from Cerulli et al. (2018). Note that the given MILP model refers to a decision planning problem of a single time slot within a working day.

\[ . \]

In the above model, the objective function (3) minimizes the total distribution cost to serve all customers, given by traveling, loading, and environmental cost components, as defined in Sect. 2.1, plus the setup cost associated with the activation of each vehicle. Constraints (4) impose that a customer is served exactly once by a vehicle, while constraints (5) impose that if a vehicle delivers goods to a customer, it must depart from it.

Constraints (6) are the so-called flow-commodity constraints, specifying the difference between the quantity of
goods a vehicle carries before and after visiting a node. Constraints (7) guarantee that the capacity of the selected vehicle is never exceeded. Constraints (8) impose that the value $y_{ijv}$ must be at least equal to demand $q_j$ of node $i$ if a vehicle $v$ crosses the arc $(i, j) \in A$. Constraints (9) ensure that the maximum length of the route traveled by each vehicle $v$ is no longer than a given value $L_v$. Finally, (10) and (11) define the decision variables.

To reduce the symmetry of the above model (3)–(11), for each class of vehicles $t$ we added the following family of constraints (12) to the model. Suppose that in a given class there are $h$ vehicles $\{v_1, v_2, \ldots, v_h\}$. For this class of vehicles we added $(h - 1)$ anti-symmetry constraints. The aim is to avoid the generation of identical solutions which differ in the allocation of routes to different vehicles of the same class. In particular, to create these constraints, we associate a unique integer number $id(d, j)$ to each outgoing arc from the depot.

$$\sum_{(d,j) \in A} id(d, j)x_{dijv} \geq \sum_{(d,j) \in A} id(d, j)x_{dijv}(i+1) \forall i \in [1, h]$$

Constraint (12) allows us to create a route/vehicle assignment order. The ordering ensures that in the case of $h$ equivalent vehicles, the route vehicle assignment will always be unique. This is true because thanks to our reduction algorithm (see Sect. 2.2), the arc leaving the depot is associated with the first customer served on the route, making the integer $id(d, j)$ unique for each route.

4 Heuristic approach

In our study, we present two different approaches to solve our GVRP problem: a math-heuristic approach and a memetic approach.

4.1 Math-heuristic approach

The math-heuristic methods proposed in the literature to solve a different variant of the VRP could be classified in the following three classes (see the survey by Archetti and Speranza 2014).

- Decomposition approaches: the problem is split into smaller and simpler subproblems, and a specific solution method is applied to each subproblem.
- Improvement heuristics: use a mathematical programming model to improve a solution found by a different heuristic approach.
- Column generation-based approaches: Branch-and-price algorithms. Such algorithms use a set partitioning formulation by associating a binary variable to each possible route (column). Due to the exponential number of variables, the solution of the linear relaxation of the formulation is performed through column generation.

Our approach can be considered belonging to class 2 with a substantial modification: we designed the heuristic in order to give more importance to the optimization of the objective function at the expense of its feasibility. The restoration of the feasibility is entrusted to the mathematical model. In detail, our math-heuristic (MAT) approach uses the mathematical model to solve a “reduced” version of the instance to be solved.

For each instance to be solved, we perform the iterated local search (ILS) technique several times, using a multi-start approach, and analyze all the different solutions obtained. As a result of this process, we try to identify how often each arc is present or not in the generated solutions. In this way, we reduce the size of the instance by eliminating arcs that have a low probability of being part of the optimal solution and leaving (promising) arcs that have a good probability of being part of the optimal solution. To implement this reduction, for each arc, we count the number of times it is present in the final ILS solution: those arcs that exceed a predetermined threshold are left in the graph, while those below this threshold are eliminated. Obviously, this reduction process is carried out by preserving the connection of the graph. In the implementation of the proposed technique, It is chosen a dynamic threshold for each node, such that at least five incoming and five outgoing arcs are present for each node of the graph after the reduction.

4.2 Memetic approach

An extensive review and analysis of the main meta-heuristics applied to the various VRP problems were presented in the paper by Vidal et al. (2013, 2019). The meta-heuristics used to solve VRP and which have given the best results are Tabu search, simulated annealing, deterministic annealing (Li et al. 2012), genetic algorithms (Minocha et al. 2011), ant colony (Donati et al. 2008), and variable neighborhood search implemented by Hemmelmayr et al. (2009).

The memetic approach we designed is obtained by embedding our simple local search (LS) in a genetic framework. In particular, we apply the LS to each chromosome that our GA produces. We use this memetic approach instead of a classic GA to guarantee the feasibility of the solutions generated. In fact, constraint (9) in the model (3)–(12), which limits the maximum length of each route, makes it very hard to get feasible solutions. For this reason, the crossover and mutation operators we designed are highly destructive and, in general, do not guarantee the feasibility of the solutions produced. Then, we combine the power of the GA that tries to improve the solutions without caring too much about the
feasibility with the accuracy of the LS that looks to restore the feasibility.

Each solution $S$ is composed of a set of routes $R$. A vehicle and a sequence of nodes representing the path are associated with each route. The first element of each route is always the depot $d$. An example of a solution to our problem is reported in Fig. 4.

In the remainder of this paper, we denote $|S|$ the number of routes contained in a solution $S$, and we use $|R|$ to indicate the number of customers in the route $R$.

4.3 Constructive algorithm

The purpose of this algorithm is to create an initial solution (not necessarily feasible) to be used as input for the successive approaches. Until all customers are not served, this algorithm: (i) creates an empty route; (ii) associates a vehicle with the route; (iii) adds other customers to the route until the vehicle has residual capacity. The route always starts from the depot. Then, iteratively, the customers are added, choosing among the not served customers one, close enough to the last customer reached. Note that this scheme does not take into account constraints (9) of the MILP model, while constraints (7) are considered in the construction phase of the solution. A detailed pseudo-code of the algorithm is shown in Algorithm 3.

This algorithm creates a random solution $S$. As it is possible to notice for the schema reported in Algorithm 3, this solution depends on the parameter $px$. More precisely, with $px = 1$, we have a solution in which, for every route, the Nearest Neighbor policy is used. Increasing the value of $px$, we can get very different solutions from each other. Since this procedure takes into account only the capacity of the vehicle and not the maximum travel time of a route, it is possible to find unfeasible solutions to the problem. It would be easy to remedy the problem, but since the solutions produced are only useful as initial solutions for the algorithms proposed in the next sections, a degree of unfeasibility has proved to be experimentally useful to guarantee a wider exploration of the feasible region.

4.4 Fitness function

In order to compare two solutions and determine the best, it is usually sufficient to use the objective function associated with the problem. Unfortunately, when algorithms have to compare solutions that are not always feasible, the use of the objective function alone is not enough. In fact, the objective function value may not be calculable in the case of an unfeasible solution, or this value may not be representative of the goodness of the solution.

In this paper, we use the objective function as a fitness function in the case of a feasible solution. In the case of an unfeasible solution, we add a penalty to the fitness function useful to represent the degree of the infeasibility of the solution.

Given the solution $S$, $S_r$ represents the objective function value of the solution. In the case of an unfeasible solution, $S_r$ represents the total violation in terms of a vehicle’s maximum capacity (violation of constraint 6), and $S_8$ represents the total violation in terms of routes length (violation of constraint 8). The fitness function $ff$ computed for solution $S$ is equal to:

$$ ff(S) = S_r + M + M_6S_8 + M_8S_8, $$

where $M$, $M_6$ and $M_8$ are

| $S_1$ | $v_1$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ |
| $S_2$ | $v_2$ | $n_6$ | $n_7$ | $n_8$ | $n_9$ |
| $S_3$ | $v_3$ | $n_{10}$ | $n_{11}$ | $n_{12}$ | $n_{13}$ | $n_{14}$ | $n_{15}$ |

Fig. 4 Representation of a generic solution

Algorithm 3 ConstructiveAlgorithm

1: Input: $G(N, A)$, $U$, $V$, $px$.
2: Create an empty solution $S$.
3: while $U \neq \emptyset$ do
4: Remove a random vehicle $v$ from $V$.
5: Create an empty route $R$.
6: Add $R$ in the solution $S$.
7: Set $v$ in $R$.
8: Add the depot $d$ to $R$.
9: while $U \neq \emptyset$ do
10: let $n$ the last node added to $R$.
11: $n_{next} = GetNextCustomer(U, v, n, px)$
12: if $k_v \geq q_1 + \sum_{i \in k} q_i$ then
13: Add $n_{next}$ to $R$.
14: Remove $n_{next}$ from $U$.
15: else
16: Exit while.
17: end if
18: end while
19: end while
20: Return $S$.
21: function GetNextCustomer
22: Input: $U$, $v$, $n$ $px$.
23: Arrange in ascending order the elements $u \in U$ according to the value $c_{nu}$.
24: return a random $u \in U$ selected in the first $px$ positions of $U$.

| $R_1$ | $v_1$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ |
| $R_2$ | $v_1$ | $n_6$ | $n_7$ | $n_8$ | $n_9$ |
| $R_3$ | $v_3$ | $n_{10}$ | $n_{11}$ | $n_{12}$ | $n_{13}$ | $n_{14}$ | $n_{15}$ |

Fig. 5 Example of neighborhood 2-OPT applied on the couple on arcs $(n_1, n_2)$, $(n_5, n_6)$

| $R_1$ | $v_1$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ |
| $R_2$ | $v_2$ | $n_6$ | $n_7$ | $n_8$ | $n_9$ |
| $R_3$ | $v_3$ | $n_{10}$ | $n_{11}$ | $n_{12}$ | $n_{13}$ | $n_{14}$ | $n_{15}$ |

Fig. 6 Example of neighborhood Swap, the customer $n_2 \in R_1$ is exchanged with the customer $n_8 \in R_2$. 
three penalty multipliers. For all three $M$ penalty parameters in the experimental tests, we used the value 10,000.00.

4.5 Local search

A local search algorithm was developed to make the solution proposed by the constructive approach feasible and improve its quality. The local search is a technique that makes use of the neighborhood of a solution. Starting from an initial solution, this is iteratively improved by exploring its neighborhood until the technique reaches a local minimum. In order to describe the implemented algorithm, it is necessary to introduce the neighborhood used. In our implementation, we used four different neighborhoods combined in sequence. Before describing the algorithm, we introduce the used neighborhood.

**Neighborhood 1: One-Change.** For each route $R \in S$, for each customer $n \in R$, $n$ is removed from $R$ and reinserted in $R$ in every possible position. This neighborhood will therefore contain a number of solutions equal to $|S| \times |R| \times (|R| - 1)$.

**Neighborhood 2: 2-OPT.** For each route $R \in S$, for each couple of arcs $(n_i, n_j), (n_k, n_h) \in R$, the arcs $(n_i, n_j)$ and $(n_k, n_h)$ are removed from $R$, the path from $n_j$ to $n_k$ in $R$ is reversed and the arcs $(n_i, n_h), (n_j, n_k)$ are added to $R$ to close the route. This neighborhood will therefore contain a number of solutions equal to $|S| \times |R| \times (|R| - 1)$. Figure 5 shows an example of this operator.

**Neighborhood 3: Swap.** For each pair of routes $R_1, R_2 \in S$, for each pair of customers $n_1 \in R_1, n_2 \in R_2$, in the route $R_1$ the customer $n_1$ is replaced by $n_2$ and in the route $R_2$ the customer $n_2$ is replaced by $n_1$. This neighborhood will therefore contain a number of solutions equal to $|S| \times (|S| - 1) \times |R_1| \times |R_2|$. Figure 6 shows an example of this operator.

**Neighborhood 4: Migration.** For each pair of routes $R_1, R_2 \in S$, for each customer $n_1 \in R_1, n_1$ is removed from $R_1$ and added to $R_2$ in every possible position. This neighborhood will therefore contain a number of solutions equal to $|S| \times (|S| - 1) \times |R_1| \times |R_2|$. The local search algorithm is summarized in the flowchart shown in Fig. 7.

4.6 Genetic algorithm

Genetic algorithms are powerful metaheuristic algorithms, initially introduced by Holland (1975) and largely used to solve complex combinatorial optimization problems. Holland was inspired by Darwin’s theory of evolution, based on the idea of natural selection. Genetic algorithms somehow reproduce the mechanisms which characterize the evolution of life forms. Basically, they simulate the evolution of a population of individuals, each of which represents a feasible solution to the problem. The algorithm, starting with an initial population, performs a certain number of iterations, producing at each step a new generation of individuals created on the basis of the previous one. In this section, we describe a genetic algorithm for our problem. First of all, we need to specify what is the form of a solution, or rather how we encode the chromosome which represents a solution. Then, we can define the fitness function and the stopping criteria and describe how the selection, crossover, and mutation operations are performed.

**The chromosome**

The purpose of the chromosome is to represent a generic solution to the problem. The genetic algorithm uses the same
representation of the solution introduced in Sect. 4 and shown in Fig. 4.

The fitness function

The fitness function is used to associate a numerical value to each chromosome. This value is essential to be able to compare the chromosomes. Usually, this function is a direct consequence of the objective function of the problem, but it can manage the unfeasibility of some solutions through an appropriate penalty. The genetic algorithm described in this work uses the fitness function presented in Sect. 4.4.

The crossover

In a genetic algorithm, the crossover is usually used to create individuals of generation \((i + 1)\) starting from some individuals of generation \(i\). Our implementation of the crossover requires 2 chromosomes called parents and produces as output a new chromosome called son. Our crossover procedure is very simple; it starts from the parent chromosomes \(C_1, C_2\) and creates the child chromosome \(C_3\) by iterating over all the nodes of the graph excluding the depot. For each node \(n_i\), the vehicle that will serve it in \(C_3\) is chosen randomly between the vehicle that serves \(n_i\) in \(C_1\) or \(C_2\). Figure 8 shows an example of input chromosomes for the crossover. Table 2 shows for each node, the vehicle that serves it in \(C_1\) and \(C_2\). Figure 9 shows the possible crossover resulting by choosing randomly from Table 2 a vehicle for each node.

The mutation

The mutation procedure used is very invasive but, at the same time, elementary to describe. For \(\sqrt{\sqrt{N}} + 1\) nodes, a new vehicle is randomly set among all those available. All routes associated with nodes for which the vehicle has been modified are shuffled. The reasons for using such a strong mutation will be explained later.

The initial population

The initial population is created by using the constructive algorithm (Algorithm 2) introduced in Sect. 4.3, setting the parameter \(px = 5\). The population is composed of 100 chromosomes. The size of the population, as well as all the other parameters of the genetic algorithm, has been chosen following an intense manual tuning phase.

The Genetic Algorithm scheme

The GA starts from an initial population \(P\) created as described above. Then, two chromosomes \(C_1, C_2\) are chosen using a tournament selection. The tournament selection “TournamentSelectionGood” requires as input the population \(P\) and an integer denoted \(el\). The procedure randomly selects a number of chromosomes from \(P\) equal to \(el\), and returns the one with the best fitness function. The next step involves creating the \(C_3\) chromosome by applying the crossover operator. The chromosome \(C_3\) is improved by applying the previously described local search algorithm. If \(C_3\) is already present in the population, this chromosome is reconstructed using the same technique already used for the creation of the initial population. If \(C_3\) is not present in the population, its fitness is compared to the median fitness of the population. If the fitness of \(C_3\) is greater than the median fitness of the population, the mutation operator is applied to chromosome \(C_3\). Otherwise, we proceed by selecting a chromosome to be removed from \(P\) to empty a slot for \(C_3\). \(C_4\) is the chromosome to be removed from the population. It is selected through the “TournamentSelectionBad” procedure which behaves extensively like the previous tournament selection except for the fact that this procedure chooses the chromosome with worst fitness value. The procedure restarts again until the stopping criterion is reached; otherwise, the chromosome in the current population whose fitness value is the best (the minimum) is returned. This genetic algorithm scheme has the characteristic of not making the population evolve for generations but of allowing a constant evolution of the current population. To forcefully introduce the concept of generation into the population, we can say that the population of generation \(i\) differs from that of generation \(i + 1\) by exactly two chromosomes. The operating scheme of GA is summarized in Fig. 10. The extreme randomness of the operators used within this genetic algorithm is due to the
presence of a very strong local search which could alternatively cause a rapid convergence of the population towards a local minimum.

5 Computational experiments

In this section, we will report the experimental results of the tests performed. For these experiments, we used input instances of three types: (i) instances from paper (Cerulli et al. 2018), (ii) instances randomly generated, (iii) instances related to the city of Genoa. All the code has been implemented in Java, using CPLEX version 12.8 as a solver (with the default setting on all parameters and without thread limit). All tests were performed on a laptop with the following hardware features: MacBook Pro, processor Intel i9 2.9GHz, 32GB RAM. For all the experiments, we set the following operating scenario: 2 h time window, vehicle speed \(40\) Km/h, 7 min service time. This scenario implied that \(L_v = 80\) Km, \(L_c = 5\) Km. The local search algorithm was iterated 100 times (the same value as the GA population), and the best objective function value and the sum of the running times of the 100 iterations is shown in the tables reported in the next subsections.

The first set of instances used for our experiments is described in the paper of Cerulli et al. (2018) and is representative of a real problem of distribution of goods within the city of Genoa. Thanks to our mathematical model, which makes use of the reduction phase of the input graph, we were able to solve optimally all the input instances in a computation time less than 1 s, compared to the minutes or hours needed for the model proposed in the original work. For this reason, we realized that in order to test the developed approach, larger instances had to be used. In the next sections, we will introduce these instances describing their characteristics.

5.1 Random instances

This new set of instances was generated randomly; the motivation is that we wanted to carry out test cases useful to validate different characteristics of our algorithms. The approach used to create the instances can be schematized as follows: (i) place all customers within a \(1000 \times 1000\) size grid randomly; (ii) the distance between customers is set equal to the respective Euclidean distance; (iii) the cost of each arc is set equal to its length multiplied by a value in the range \([0.216: 0.432]\) randomly chosen for each type of vehicle (see Fig. 1); (iv) the demand of each customer is set to a random value in the range \([1, 10]\); (v) for each type of vehicle, the capacity is set randomly so that the sum of all the capacities of all the vehicles is double compared to the total customer demand. This leads to the generation of instances for which at least half of the vehicles must be selected. Two sets of instances were generated using this algorithm, refer-
Although the GA algorithm sometimes fails to identify a feasible solution (see the cases highlighted in bold in Table 3), the proposed model (3)–(12) is able to obtain the optimal solution in less than 1 h of CPU time in scenarios with up to 4 vehicles and a maximum of 20 customers. The LS algorithm is very fast and produces good solutions but, unfortunately, it remains trapped in an unfeasible local minimum without being able to provide a solution (see the three cases having the value “No” in Table 4). Since we noticed that the proposed LS algorithm is very fast and produces good solutions in only 100 iterations, as a further test, we wanted to verify what would happen by increasing this number to a higher running time than GA. The answer is that at 100,000 iterations, the LS becomes slower than GA and produces many unfeasible solutions (12), never reaching the quality of GA’s solution. These results are summarized in Table 5.

### 5.2 Genoa instances

In the case of large instances (see Table 4), the genetic algorithm produces very good quality solutions in a maximum CPU time of about 2 min. LS is very fast but, in cases where it is very difficult to find a feasible solution, often it remains trapped in an unfeasible local minimum without being able to provide a solution (see the three cases having the value “No” in Table 4). Since we noticed that the proposed LS algorithm is very fast and produces good solutions in only 100 iterations, as a further test, we wanted to verify what would happen by increasing this number to a higher running time than GA. The answer is that at 100,000 iterations, the LS becomes slower than GA and produces many unfeasible solutions (12), never reaching the quality of GA’s solution. These results are summarized in Table 5.
Table 5  
| Iterations | Gap (%) | Time |
|------------|---------|------|
| 1000       | 2       | 13   |
| 10,000     | 2       | 132  |
| 100,000    | 1       | 132  |

Using the instances shown in Table 4

| V | CI | U | LS Value | Time | GA Value | Time | #Inf. |
|---|----|---|---------|------|----------|------|-------|
| 6 | 2  | 50| 133.4   | 0.1  | 130.5    | 29.8 |
| 8 | 2  | 60| No      | 0.1  | 150.7    | 49.0 |
| 9 | 2  | 80| No      | 0.2  | 170.6    | 107.8|
| 10| 2  | 100| No      | 0.3  | 226.7    | 120.8|
| 12| 3  |   | 187.5   | 0.1  | 169.7    | 99.4 |
| 100|  6 |    |        | 0    |          |      |
| 1000|  3 |    |        | 1    |          |      |
| 10,000| 2 |    |        | 1   |          |      |
| 100,000|1|    |        | 132 |          |      |

Refer to Table 3 for the meaning of the columns
city of Genoa allows us to say that the proposed algorithms are a valuable aid in the definition of sustainable distribution plans in urban areas, also considering that the proposed model is easily adaptable to different urban realities and mobility rules.

In the future, we would like to investigate the proposed math-heuristic approach further. In particular, we want to make the mathematical model “tighter” and efficient and want to deepen the ways of reducing the size of the input graph based on the solutions proposed by the local search procedure.

**Funding** Open access funding provided by Università degli Studi di Genova within the CRUI-CARE Agreement. The authors did not receive support from any organization for the submitted work.

**Data Availability** The datasets analyzed during the current work are available from the corresponding author on reasonable request.

**Conflict of interest** The authors declare that they have no conflict of interest.

**Ethics approval** This article does not contain any studies with human participants or animals performed by any of the authors.

**Code availability** Code developed for the model will be made available from the corresponding author on reasonable request.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

**Declarations**
References

Archetti C, Speranza MG (2014) A survey on matheuristics for routing problems. EURO J Comput Optim 2(4):223–246
Behnke M, Kirschstein T (2017) The impact of path selection on ghg emissions in city logistics. Transp Res E Logist Transp Rev 106:320–336
Bektas T, Laporte G (2011) The pollution-routing problem. Transp Res B Methodol 45(8):1232–1250
Bektas T, Ehmke JF, Psarafitis HN, Puchinger J (2019) The role of operational research in green freight transportation. Eur J Oper Res 274(3):807–823
Bigazzi AY, Figliozzi MA (2013) Marginal costs of freeway traffic congestion with on-road pollution exposure externality. Transp Res A Policy Pract 57:12–24
Capobianco G, Cerrone C, Cerulli R, Felici G (2017) Optimal paths for dual propulsion vehicles on real street network graphs. In: International conference on optimization and decision science. Springer, pp 517–527
Carrabs F, Cerulli R, Sciomachen A (2014) In: Dameri RP, Rosenthal-Sabroux C (eds) Environmental sustainable fleet planning in B2C e-Commerce urban distribution networks. Springer, Cham, pp 183–192
Carrabs F, Cerulli R, Sciomachen A (2017) An exact approach for the grocery delivery problem in urban areas. Soft Comput 21 (9):2439–2450
Cerrone C, Dussault B, Wang X, Golden B, Wasil E (2019) A two-stage solution approach for the directed rural postman problem with turn penalties. Eur J Oper Res w272(2):754–765
Cerrone C, Cerulli R, Sciomachen A (2021) Grocery distribution plans in urban networks with street crossing penalties. Networks 78(3):248–263
Cerulli R, Dameri RP, Sciomachen A (2018) Operations management in distribution networks within a smart city framework.IMA J Manag Math 29(2):189–205
Chang T-H, Tseng J-S, Hsieh T-H, Hsu Y-T, Lu Y-C (2018) Green transportation implementation through distance-based road pricing. Transp Res A Policy Pract 111:53–64
Chen D, Pan S, Chen Q, Liu J (2020) Vehicle routing problem of contactless joint distribution service during covid-19 pandemic. Transp Res Interdiscip Perspect 8:100233
Dekker R, Bloemhof J, Mallidis I (2012) Operations research for green logistics—an overview of aspects, issues, contributions and challenges. Eur J Oper Res 219(3):671–679
Demir E, Bektas T, Laporte G (2014) The bi-objective pollution-routing problem. Eur J Oper Res 232(3):464–478
Donati AV, Montemanni R, Casagrande N, Rizzoli AE, Gambardella LM (2008) Time dependent vehicle routing problem with a multi ant colony system. Eur J Oper Res 185(3):1174–1191
Ehmke JF, Campbell AM, Thomas BW (2018) Optimizing for total costs in vehicle routing in urban areas. Transp Res E Logist Transp Rev 116:242–265
Erdoğan S, Miller-Hooks E (2012) A green vehicle routing problem. Transp Res E Logist Transp Rev 48(1):100–114
Figliozzi MA (2010) The impacts of congestion on commercial vehicle tour characteristics and costs. Transp Res E Logist Transp Rev 46(4):496–506
Hemmelmayr VC, Doerner KF, Hartl RF (2009) A variable neighborhood search heuristic for periodic routing problems. Eur J Oper Res 195(3):791–802
Heshmati S, Verstichel J, Esprit E, Berghe GV (2019) Alternative e-commerce delivery policies: a case study concerning the effects on carbon emissions. EURO J Transp Logist 8(3):217–248
Holland J (1975) Adaptation in nature and artificial systems. University of Michigan Press, Ann Arbor
Kauf S (2019) Smart logistics as a basis for the development of the smart city. Transp Res Procedia 39:143–149
Kellner F, Otto A, Braband C (2017) Bringing infrastructure into pricing in road freight transport: a measuring concept based on navigation service data. Transp Res Procedia 25:794–805
Koyuncu I, Yavuz M (2019) Duplicating nodes or arcs in green vehicle routing: a computational comparison of two formulations. Transp Res E Logist Transp Rev 122:605–623
Kramer R, Subramanian A, Vidal T, Lucidio dos Anjos FC (2015) A matheuristic approach for the pollution-routing problem. Eur J Oper Res 243(2):523–539
Li X, Leung SC, Tian P (2012) A multistart adaptive memory-based Tabu search algorithm for the heterogeneous fixed fleet open vehicle routing problem. Expert Syst Appl 39(1):365–374
Minocha B, Tripathi S, Mohan C (2011) Solving vehicle routing and scheduling problems using hybrid genetic algorithm. In: 2011 3rd International Conference on Electronics Computer Technology, vol 2. IEEE, pp 189–193
Okushima M (2018) DIntegrated network transport simulator to evaluate transport policy for reduction of carbon dioxide emission. Transp Res Procedia 34:283–290
OpenStreetMap contributors: planet dump retrieved from https://planet.osm.org. https://www.openstreetmap.org (2017)
Petro F, Konečný V (2019) Calculation of external costs from production of direct and indirect emissions from traffic operation. Transp Res Procedia 40:1162–1167
Ricardo A (2014) Final report for the European Commission. Report ED57769. European Commission—Mobility and Transport DG, Brussels, p 28-0361049
Strale M (2019) Sustainable urban logistics: What are we talking about? Transp Res A Policy Pract 130:745–751
Szymczyk K, Kadlubek M (2019) Challenges in general cargo distribution strategy in urban logistics—comparative analysis of the biggest logistics operators in EU. Transp Res Procedia 39:525–533
Taniguchi E, Thompson RG, Yamada T (2012) Emerging techniques for enhancing the practical application of city logistics models. Procedia Soc Behav Sci 39:3–18
Vidal T, Crainic TG, Gendreau M, Prins C (2013) Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. Eur J Oper Res 231(1):1–21
Vidal T, Laporte G, Matl P (2019) A concise guide to existing and emerging vehicle routing problem variants. Eur J Oper Res 6:66
Xu X, Aziz HA, Guensler R (2019) A modal-based approach for estimating electric vehicle energy consumption in transportation networks. Transp Res D Transp Environ 75:249–264
Yu Y, Wang S, Wang J, Huang M (2019) A branch-and-price algorithm for the heterogeneous fleet green vehicle routing problem with time windows. Transp Res B Methodol 122:511–527

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.