A Dynamic Multi-Period General Routing Problem Arising in Postal Service and Parcel Delivery Systems

Demetrio Laganà[*], Gilbert Laporte[†], Francesca Vocaturo[§]

Abstract

Postal and courier companies are under the pressure of minimizing operational costs while meeting requests for increasingly high service levels. We model their delivery problem as a dynamic multi-period general routing problem (DMPGRP) with the aim of minimizing the total cost over a given planning horizon. In most modern delivery systems, fluctuating demand volumes dynamically reveal themselves over time; in addition, service differentiation is highly important. Both these characteristics are considered in our DMPGRP. An effective solution strategy for this problem selects demands that must be fulfilled day by day, after distinguishing them according to priority classes. It then constructs the vehicle routes by using a heuristic algorithm based on the adaptive large neighbourhood search. The results of an extensive computational study are reported.

Keywords. *Delivery service; mixed capacitated general routing problem; adaptive large neighbourhood search*

1 Introduction

The delivery of parcels, packages, and letters has become an essential part of the transportation industry, as emphasized by Kovacs et al. [20]. The same authors point out that the

[*]Department of Mechanical, Energy and Management Engineering, University of Calabria, 87036 Arcavacata di Rende (CS), Italy. E-mail address: demetrio.lagana@unical.it

[†]CIRRELT and Canada Research Chair in Distribution Management - HEC Montréal, Montréal H3T 2A7, Canada. E-mail address: gilbert.laporte@cirrelt.ca.
School of Management, University of Bath, Bath BA2 7AY, United Kingdom.

[§]Department of Economics, Statistics and Finance "Giovanni Anania", University of Calabria, 87036 Arcavacata di Rende (CS), Italy. E-mail address: vocaturo@unical.it

competition in small package shipping via carriers and couriers is intense. Consequently, private (e.g., DHL, FedEx,UPS) and public companies (e.g., national postal services) have to operate at maximum efficiency to remain in this market. The market that benefits from the boom in e-commerce is ever growing [14].

In this context, several delivery systems have reinterpreted their original mission in a modern light. An interesting example arises in postal service organizations that have gone through substantial changes in order to address the decrease of handwritten letters, bills, invoices, publicity material, etc. Yet they continue to deliver these items, whose volume has decreased but not disappeared, in addition to managing the traditional flow of postal parcels for their own customers. Quite significantly, in most countries postal service organizations have started to focus on a new business by acting as third party logistics providers for on-line market places. In effect, they already possess infrastructures, means of transportation and human resources to face challenges and exploit new opportunities in the fast-delivery market. For instance, Poste Italiane focuses on parcel delivery services (Express Courier, Logistics, and Parcels business), which have been expanding constantly in terms of volumes and revenues. Notably, in June 2018, Poste Italiane forged a new three-year partnership with Amazon (renewable for an additional two years) for the delivery of e-commerce packages [25].

Note that, as in any other competitive and mature market, opportunities for service differentiation also exist in the parcel delivery market generated by e-commerce. Premium service offerings attempt to cater for the needs of customers requiring shorter shipping times and are already proposed by many on-line marketplaces [17]. An example is given by Amazon Prime service.

Now more than ever, to improve the efficiency of postal service and parcel delivery systems, the main research stream is to apply operations research approaches with the aim of optimizing transport operations, e.g., distribution network design, transport planning, and routing. In this context, we focus on routing and we model the delivery problem as a *dynamic* decision problem over a discrete planning horizon consisting of several days. A decision problem is said to be *static* when all the input data of the problem are known before decisions are made. In contrast, in a dynamic environment, some of the input data are revealed or updated during the period of time in which the operations linked to the decisions take place. The planning horizon of a dynamic problem may be unbounded, in which case the solution to the problem cannot be a standard output, but rather a strategy which, using the revealed information, specifies the actions that must be carried out as time goes by [10].

More specifically, in our reference context, there are items that must be delivered to specified locations (recipients). We assume that the set of possible recipients is given. This assumption is realistic, but it is unrealistic to suppose that daily demand volumes and service request deadlines are known several days earlier. In effect, in many real-world applications, the demands are fluctuating and unpredictable in the short term for both traffic volumes and points where daily recipients are located, despite the presence of highly equipped analytics departments. This may occur especially in e-commerce contexts (e.g., unpredictable peaks can be induced by special promotions).

1.1 Related Literature

Resorting to robust optimization models [8] does not seem to constitute a feasible option in the case tackled in this article in which no information about the future demand is available, not even bound values. Hence, an appropriate solution framework seems to be a dynamic multi-period model.

An interesting study about a dynamic multi-period vehicle routing problem is the one proposed by Wen et al. [31] who solved the case of Lantmännen, a large Swedish distributor in the food, energy and agricultural industries. In their study, the authors consider customers that place orders over a planning horizon. The orders are revealed incrementally over time, i.e., without knowledge of future demand quantities from the distributor. Each request also specifies a delivery location and a set of consecutive periods (or days) during which delivery can take place. A fleet of homogeneous vehicles is used in the distribution activity. The objectives are to minimize the total routing cost and customer waiting, and to balance the daily work load over the planning horizon. A similar problem was tackled by Angelelli et al. [3] who considered companies that receive orders each day and have the flexibility to fulfill them within the two days after the orders are received. A single vehicle is used in the daily distribution activity. Decisions have to be made on the basis of partial information with the aim of minimizing the total travel cost over time. In effect, customer demands are not known in advance but, also in this case, become available incrementally over the planning horizon. The study just described was extended by Angelelli et al. [2] in a context of systems dealing with on-line pickup requests. We also mention an extension of the dynamic multi-period vehicle routing problem known as the dynamic multi-period vehicle routing problem with probabilistic information. In this type of problem, at each time period of a given planning horizon, the set of customers requiring a service in later time periods is unknown, but its probability distribution is available [1]. A similar problem was analysed by Ulmer et al. [28] who formulated a dynamic multi-period vehicle routing problem with stochastic service requests as a route-based decision Markov process and presented an anticipatory policy based on approximate dynamic programming.

Other interesting scientific contributions related to our study are here briefly described. Azi et al. [6] studied a delivery problem in which the service requests are not known in advance and the vehicles execute multiple routes during their workday. In this dynamic problem, answers to new requests (received according to independent time-space Poisson processes) must to be provided in real-time. Obviously, the decisions are based on the planned routes to be executed later (thus, excluding the current route). Pérez Rivera and Mes [24] investigated the decision problem of selecting freights for transportation in long-haul round-trips. In this problem, freight demands become known gradually over time and have to be delivered and picked up at different locations. Although the number of demands and their characteristics vary from day to day, there is information about their probability distribution. The problem was modelled as a Markov decision process and solved through an approximate dynamic programming algorithm. Approximate dynamic programming approaches were also proposed by van Heeswijk et al. [29] and Zhang et al. [32] to tackle a dynamic dispatching problem and a dynamic orienteering problem, respectively. In particular, van Heeswijk et al. [29] dealt with large instances of the decision problem faced by an urban consolidation centre which dynamically receives customer orders and dispatches them in batches for the last-mile distribution (the orders have a known associated probability function). Zhang et al. dealt with a decision problem where a traveller must arrive and provide service at locations within the respective time windows to collect rewards. At each location, the traveller must wait in a queue which consists of the competitors; the queue length is unknown to the traveller before arrival and the wait time is stochastic. More recently, Dayarian and Savelsbergh [16] have studied a same-day home delivery problem with crowdshipping (a new form of sharing or collaborative economy) by also considering a dynamic variant where no information or distributional information is available.

Finally, we mention the periodic routing problem where decisions have to be made over a given planning horizon in a static environment. There exists a vast body of literature addressing both interesting applications and tailored solution methods for this problem and its extensions [13]. The article of Benavent et al. [9] constitutes a very recent reference on the topic. Specifically, the authors presented a branch-and-cut algorithm to optimally solve a periodic routing problem with irregular service requests. Such a problem arises in applications related to road maintenance and road network surveillance in an arc routing context. The literature proposes other classes of static routing problems dealing with multi-period plans. For instance, Archetti et al. [5] studied the multi-period vehicle routing problem with due dates, where customer orders have to be fulfilled between a release date and a due date (if the due date of an order exceeds the planning horizon, its service may be postponed by charging a penalty). The authors investigated and computationally compared three alternative formulations of the problem.

1.2 Contributions and Article Structure

This article emphasizes the need to investigate aspects of service differentiation when developing decision support instruments. In particular, our main scientific contribution is to show how even simple solution schemes considering service differentiation actually outperform the approaches traditionally proposed in the scientific literature for planning vehicle routes in delivery systems like e-commerce third-party logistics operators and modern postal service organizations. In addition, we are concerned about dealing with unpredictable demands in the short term. The value of anticipating future events is pointed out in many scientific articles (see, e.g., [16]) and is relevant to us. However, it is also common to operate under incomplete knowledge.

To this end, we consider a multi-period routing problem in a highly dynamic setting. We propose two solution strategies, called greedy and spread. Through an extensive computational study we show that the spread strategy, which considers service differentiation, outperforms the greedy strategy, which ignores it. As a consequence, our experiments confirm the value of flexibility in planning deliveries.

The remainder of the article is organized as follows. A more detailed description of our routing problem is given in the next section. The section also introduces our notation. Section 3 describes the two solution strategies mentioned above. Section 4 proposes a heuristic algorithm to tackle the delivery problem through the spread strategy. Section 5 illustrates the outputs of an extensive computational phase. Conclusions follow in Section 6.

2 Problem Description

Our contribution is of general applicability and can be adapted to many postal and distribution organizations that face a similar problem. To model the distribution framework, we use a mixed general routing structure. This structure, which uses a mixed graph to represent the street networks, allows us to represent isolated users (or entities receiving many items like large companies) as vertices of the graph, and group of common recipients (e.g., private households on a same street) as links, i.e. (directed) arcs or (undirected) edges. General routing problems arise in classic arc routing contexts. More effectively than other routing problems, they can model real-world applications in urban waste collection, post and parcel distribution, school bus routing, etc. [15]. For a recent literature review on general routing problems the reader is refereed to the annotated bibliography of Mourão and Pinto [23]. We assume that if a group of recipients is modelled as a link, then it is always handled as a unit, even when only one of the recipients must be serviced. Therefore, the graph underlying the problem does not change over the time.

In most real systems, the items are delivered with a speed related to their nature or, more frequently, to the desired speed of delivery. It is therefore necessary for the delivery operator to introduce *priority classes* and suppose that some deliveries can be postponed.

2.1 Representation of Differentiated Services in a Dynamic Setting

For the purpose of modeling, we consider a discrete planning horizon H which can be bounded or unbounded. On each day $h \in H$, new items arrive at a specific vertex of the graph called the *depot*. This vertex represents a central hub at which a fleet of homogeneous vehicles used in delivery operations is based. The total volume or weight associated with all items addressed to the same recipient (vertex) or to the same group of recipients (link) define its *service demand* for day h. In order to simplify the discussion, we consider only three priority classes for the demands, i.e., *urgent*, *prominent*, and *unimportant*: (i) an urgent demand includes at least one item that must be delivered quickly, like a parcel associated with a 24-hour express service; (ii) a prominent demand includes at least one item that must be delivered within two days; (*iii*) an unimportant demand only includes items of different nature with respect the urgent and prominent ones. On day h, an urgent demand has to be totally fulfilled (i.e., all items defining it have to be delivered) through the delivery routes planned for that day. Deliveries associated with prominent and unimportant demands can be postponed. We do not consider partial fulfilments. Therefore, prominent and unimportant demands can be totally fulfilled or totally unfulfilled on day h. Unfulfilled demands constitute the so-called *outstanding work* and are reconsidered on day $h+1$. In particular, they are added to the new demands that arise at the depot on day $h+1$ (Figure 1). The priorities of the unfulfilled demands accordingly change over time. We assume

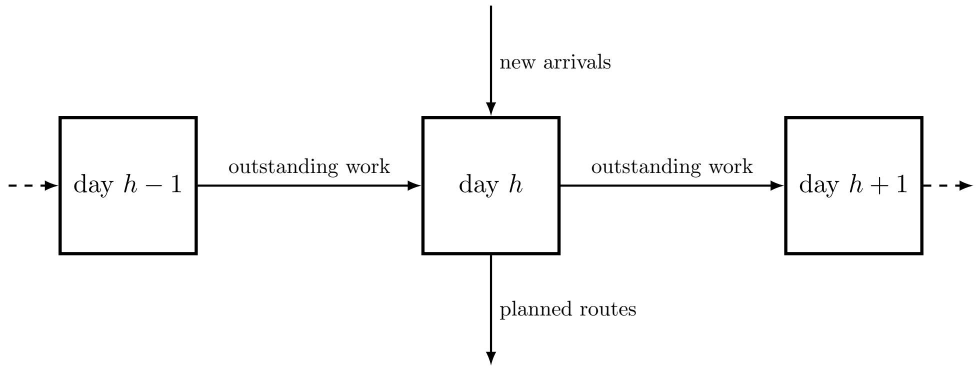

Figure 1: Simplified operation scheme

that the demands classified as unimportant become prominent after two days if they have been unfulfilled, while the demands classified as prominent become urgent after one day. Obviously, whenever old and new demands are mixed, the class with the higher priority prevails. For instance, if an unfulfilled prominent demand on day h is added to a new unimportant demand on day $h+1$, then the resulting *aggregate demand* for day $h+1$ is defined by their sum and is classified as urgent. Note that it is easy to modify the problem and the methods presented in this article if a different number of priority classes and different rules to change priorities over time are considered.

The key element of our problem is that decisions must be made each day without any knowledge of future demands. There are neither probability distributions nor bounds and not even approximate values. The problem just described is defined as the *Dynamic Multi-Period General Routing Problem* (DMPGRP). The problem consists of designing daily vehicle routes satisfying the service requirements and respecting the delivery priorities.

Each day, the set of planned routes has to be consistent with the number of available vehicles and their capacity. The aim is to minimize the average routing cost per demand unit.

We assume that new arrivals arise before planning routes for the current day (in practical situations, the planning horizon is almost unbounded and arrivals after a given hour are treated for the first time on the next day). This feature distinguishes the DMPGRP from other problems in which decision makers can immediately deal with the available demands when vehicles are already moving in the area of interest. In addition, we impose that all demands be fulfilled within the planning horizon. Thus, if the horizon is bounded, all demands defined on the last day have to be fulfilled independently of their priority level. In other words, in any feasible solution the total demand arising within a bounded planning horizon must be fulfilled within that horizon. This means that different solution approaches to the DMPGRP can be fairly compared with each other. Note that, if the planning horizon is bounded, minimizing the average routing cost per demand unit corresponds to minimizing the total routing cost over that horizon.

2.2 Notation

We introduce the input data for the DMPGRP. The notation reported in this section will be also used in the mathematical formulation presented later. We have already defined H as a discrete planning horizon. Let $G = (V, A, E)$ be a mixed graph defined by a set of vertices V, a set of arcs A, and a set of edges E. Vertex $1 \in V$ represents the depot at which a set K of identical vehicles of capacity Q are based. The set of vertices different from the depot is denoted by $C = V \backslash \{1\}$. Any link of $A \cup E$ can be deadheaded (i.e., traversed without service) any number of times. The traversal of link $(i, j) \in A \cup E$ yields a non-negative traversal cost c_{ij}. Let $\hat{A}$ and $\hat{E}$ be the sets of arcs and edges that must be serviced at least once over the planning horizon. Let $u_{ij}^h \geq 0$ be the value of an urgent demand received on day $h \in H$ and addressed to link $(i, j) \in \hat{A} \cup \hat{E}$, $p_{ij}^h \geq 0$ the value of a prominent demand received on day $h \in H$ and addressed to link $(i, j) \in \hat{A} \cup \hat{E}$, and $t_{ij}^h \geq 0$ the value of an unimportant demand received on day $h \in H$ and addressed to link $(i, j) \in \hat{A} \cup \hat{E}$. Note that for any link $(i, j) \in \hat{A} \cup \hat{E}$, at most one of u_{ij}^h, p_{ij}^h, and t_{ij}^h can be positive. In effect, a non-zero demand received on day $h \in H$ and addressed to a link $(i, j) \in \hat{A} \cup \hat{E}$ has only one status: urgent, prominent or unimportant. In addition, in our dynamic setting, the values associated with u_{ij}^h, p_{ij}^h, and t_{ij}^h become known starting from day $h \in H$.

Let $\hat{V} \subseteq C$ be the set of vertices that must be serviced at least once over the planning horizon. Let $u_i^h \geq 0$ be the value of an urgent demand received on day $h \in H$ and addressed to vertex $i \in \hat{V}$, $p_i^h \geq 0$ the value of a prominent demand received on day $h \in H$ and addressed to vertex $i \in \hat{V}$, and $t_i^h \geq 0$ the value of an unimportant demand received on day $h \in H$ and addressed to vertex $i \in \hat{V}$. Similarly, for any vertex $i \in \hat{V}$, at most one of u_i^h, p_i^h, and t_i^h can be positive. The values associated with u_i^h, p_i^h, and t_i^h become known starting from day $h \in H$.

An element in $\hat{V} \cup \hat{A} \cup \hat{E}$ will be generically defined as a customer whenever there is no need for further specification.

3 Solution Methodology

From a methodological prospective, solving the DMPGRP means dealing with a double decision process. The first decision is to select, for each day h, the customers to be serviced on that day, and the second decision is to define the delivery routes for each day of the planning horizon. Note that, even if the overall problem is dynamic, the decision process at the beginning of day h can be viewed as static since in complete absence of information on future arrivals, the routes for that day are planned simply on the basis of the demand received (and fulfilled) so far and the routes are fixed before their execution.

3.1 Greedy Strategy

The simplest way to solve the DMPGRP is to fulfil every demand immediately, regardless of its priority. In other words, any link $(i, j) \in \hat{A} \cup \hat{E}$ with $u_{ij}^h + p_{ij}^h + t_{ij}^h > 0$ will be serviced by a delivery route planned for day h. Similarly, any vertex $i \in \hat{V}$ with $u_i^h + p_i^h + t_i^h > 0$ will be serviced by a delivery route planned for day h. We refer to this strategy as a *greedy strategy*. It ignores the outstanding work and, de facto, does not deal with service differentiation. The greedy strategy is consistent with most optimization approaches proposed in the scientific literature. In effect, within the greedy strategy, the problem of selecting the customers to be serviced day by day does not exist since it is solved at the source. Then, for each day h of the planning horizon, the routing problem can be solved by applying, without any modification, an exact or an approximate algorithm for the mixed capacitated general routing problem (see, e.g., [7, 11, 12, 19]).

3.2 Spread Strategy

In the following we will present a strategy which tackles the DMPGRP by actually considering the demand priorities. This strategy, called *spread strategy*, solves a delivery problem for every day of the planning horizon. In particular, it designs no route on day h if there is no customer associated with an urgent demand for that day. In contrast, it plans for day h the routes visiting all customers associated with an urgent demand and tries to include in them those customers associated with a prominent or an unimportant demand if it is convenient to do. The convenience is linked with the extra cost for their service. We will later present a mathematical model associated with day h. In addition to the notation defined in Section 2.2, further notation is introduced as follows.

Some subsets of $\hat{A}$ and $\hat{E}$, denoted respectively by A_h and E_h, include respectively the arcs and edges associated with a positive demand on day $h \in H$. This demand can include outstanding work from previous days or new arrivals on day h and is defined as *aggregate*. In particular, let d_{ij}^h be the aggregate demand associated with link $(i, j) \in A_h \cup E_h$. Since d_{ij}^h can include outstanding work, it does not in general coincide with $u_{ij}^h + p_{ij}^h + t_{ij}^h$. We can distinguish between the elements in A_h and E_h on the basis of the nature of the aggregate demand. In particular, we denote by A_h^R and E_h^R the subsets of arcs and edges with an urgent aggregate demand on day h, respectively. We refer to the elements in $A_h^R \cup E_h^R$ as required customers (more specifically, *required links*) for day h. Analogously, we denote by A_h^P and E_h^P the subsets of arcs and edges with a prominent aggregate demand on day h, respectively. We refer to the elements in $A_h^P \cup E_h^P$ as potential customers of level I (more specifically, *potential links of level I*) for day h. Finally, we denote by A_h^T and E_h^T, respectively, the subsets of arcs and edges with an unimportant aggregate demand on day h. We refer to the elements of $A_h^T \cup E_h^T$ as potential customers of level II (more specifically, *potential links of level II*) for day h. We have $A_h = A_h^R \cup A_h^P \cup A_h^T$ and $E_h = E_h^R \cup E_h^P \cup E_h^T$.

Similarly, the subset $V_h \subseteq \hat{V}$ includes vertices associated with a positive aggregate demand on day $h \in H$. In particular, let d_i^h be the aggregate demand associated with vertex $i \in V_h$. Subset V_h^R includes other required customers (more specifically, *required vertices*) for day h. These are the vertices associated with an urgent aggregate demand. The subsets

V_h^P and V_h^T include other potential customers of level I and II (more specifically, *potential vertices of level I and II*) for day h, respectively. These are the vertices associated with a prominent and unimportant aggregate demand, respectively. We have $V_h = V_h^R \cup V_h^P \cup V_h^T$.

Note that the elements of $A_h^R \cup E_h^R \cup V_h^R$ must be serviced on day h, whereas, the elements of $A_h^P \cup E_h^P \cup V_h^P$ and $A_h^T \cup E_h^T \cup V_h^T$ can be serviced on day h. In any case, an element cannot be split when it is serviced.

Given a subset $S \subset V$ of vertices, let $\delta^+(S)$ be the set of arcs leaving S, $\delta^-(S)$ the set of arcs entering S, $\delta_h^+(S)$ the set of required and potential arcs for day h leaving S, $\delta_h^-(S)$ the set of required and potential arcs for day h entering S, $\delta(S)$ the set of edges incident to S, and $\delta_h(S)$ the set of required and potential edges for day h incident to S. Whenever $S = \{v\}$ we replace S with v in the previous notation. Moreover, let $V_h(S)$ be the set of required and potential vertices for day h belonging to S, $A_h(S)$ the set of required and potential arcs for day h with both endpoints in S, and $E_h(S)$ the set of required and potential edges for day h with both endpoints in S.

For a link $(i,j) \in A_h \cup E_h$ and a vehicle k, let x_{ij}^k be a binary variable equal to 1 if and only if (i,j) is serviced by vehicle k which travels from vertex i to vertex j. For a link $(i,j) \in A \cup E$ and a vehicle k, let y_{ij}^k be a non-negative variable representing the number of deadheading from vertex i to vertex j by k. Finally, for a vertex $i \in V_h$ and a vehicle k, let z_i^k be a binary variable equal to 1 if and only if i is serviced by k. For simplification issue, we do not use an index h in the decision variables (it is obvious that their value refers to the day for which the model is solved, i.e., day h).

In addition, we define

$$\lambda_1 = \sum_{k\in K}\sum_{(i,j)\in E_h} c_{ij}(x_{ij}^k + x_{ji}^k) + \sum_{k\in K}\sum_{(i,j)\in A_h} c_{ij}x_{ij}^k + \sum_{k\in K}\sum_{(i,j)\in E} c_{ij}(y_{ij}^k + y_{ji}^k) + \sum_{k\in K}\sum_{(i,j)\in A} c_{ij}y_{ij}^k,$$

$$\lambda_2 = \alpha\left(\sum_{k\in K}\sum_{(i,j)\in E_h^P}\left(x_{ij}^k + x_{ji}^k\right) + \sum_{k\in K}\sum_{(i,j)\in A_h^P} x_{ij}^k + \sum_{k\in K}\sum_{i\in V_h^P} z_i^k\right),$$

$$\lambda_3 = \beta\left(\sum_{k\in K}\sum_{(i,j)\in E_h^T}\left(x_{ij}^k + x_{ji}^k\right) + \sum_{k\in K}\sum_{(i,j)\in A_h^T} x_{ij}^k + \sum_{k\in K}\sum_{i\in V_h^T} z_i^k\right),$$

where α and β are parameters concerning the inclusion of potential customers in some route, respectively for levels I and II ($\alpha > \beta$). They are used to define a *profit*. For insights, the reader is referred to the scientific literature on routing problems with profits (see, e.g., [4]). Introducing a profit leads to evaluate the degree of urgency and convenience of servicing a potential customer on day h. Specifically, on the one hand, every serviced potential customer contributes to increments the total routing cost (λ_1), on the other hand, it generates a saving equal to α for level I, to β for level II. The total saving is represented by the sum $\lambda_2 + \lambda_3$. The values of α and β play a central role in the mathematical model. If $\alpha = \beta = 0$ and an optimum exists, then there is an optimal solution where no potential customer for the current day is serviced (other optimal solutions can include the service of potential customers with extra routing cost equal to zero). Conversely, if $\alpha = \beta = M$ (where M is a large constant with respect to the traversal costs), then all potential customers for the current day will be serviced in an optimal solution compatible with vehicle capacity. Intermediate and opportunely defined values of α and β allow servicing only convenient potential customers.

The problem of selecting customers and determining vehicle routes in delivery systems on day h can be formulated as follows:

$$\text{Minimize } \lambda = \lambda_1 - \lambda_2 - \lambda_3 \tag{1}$$

subject to

$$\sum_{k \in K} (x_{ij}^k + x_{ji}^k) = 1 \qquad (i,j) \in E_h^R \tag{2}$$

$$\sum_{k \in K} x_{ij}^k = 1 \qquad (i,j) \in A_h^R \tag{3}$$

$$\sum_{k \in K} z_i^k = 1 \qquad i \in V_h^R \tag{4}$$

$$\sum_{k \in K} (x_{ij}^k + x_{ji}^k) \leq 1 \qquad (i,j) \in E_h^P \cup E_h^T \tag{5}$$

$$\sum_{k \in K} x_{ij}^k \leq 1 \qquad (i,j) \in A_h^P \cup A_h^T \tag{6}$$

$$\sum_{k \in K} z_i^k \leq 1 \qquad i \in V_h^P \cup V_h^T \tag{7}$$

$$\sum_{(i,j) \in E_h} d_{ij}^h (x_{ij}^k + x_{ji}^k) + \sum_{(i,j) \in A_h} d_{ij}^h x_{ij}^k + \sum_{i \in V_h} d_i^h z_i^k \leq Q \qquad k \in K \tag{8}$$

$$\sum_{j:(i,j)\in\delta_h^+(i)} x_{ij}^k + \sum_{j:(i,j)\in\delta^+(i)} y_{ij}^k - \sum_{j:(j,i)\in\delta_h^-(i)} x_{ji}^k - \sum_{j:(j,i)\in\delta^-(i)} y_{ji}^k =$$

$$= \sum_{j:(i,j)\in\delta_h(i)} x_{ji}^k + \sum_{j:(i,j)\in\delta(i)} y_{ji}^k - \sum_{j:(i,j)\in\delta_h(i)} x_{ij}^k - \sum_{j:(i,j)\in\delta(i)} y_{ij}^k \qquad k \in K,\ i \in V \quad (9)$$

$$\sum_{(i,j)\in\delta_h^+(S)} x_{ij}^k + \sum_{(j,i)\in\delta_h^-(S)} x_{ji}^k + \sum_{(i,j)\in\delta_h(S)} (x_{ij}^k + x_{ji}^k) + \sum_{(i,j)\in\delta^+(S)} y_{ij}^k +$$

$$+ \sum_{(j,i)\in\delta^-(S)} y_{ji}^k + \sum_{(i,j)\in\delta(S)} (y_{ij}^k + y_{ji}^k) \geq \begin{cases} 2(x_{uv}^k + x_{vu}^k) & (u,v) \in E_h(S), \\ 2x_{uv}^k & (u,v) \in A_h(S), \\ 2z_u^k & u \in V_h(S), \end{cases}$$

$$k \in K,\ S \subseteq C \quad (10)$$

$$x_{ij}^k \in \{0,1\} \qquad k \in K,\ (i,j) \in A_h \cup E_h \quad (11)$$

$$x_{ji}^k \in \{0,1\} \qquad k \in K,\ (i,j) \in E_h \quad (12)$$

$$y_{ij}^k \in \mathcal{Z}_+ \qquad k \in K,\ (i,j) \in A \cup E \quad (13)$$

$$y_{ji}^k \in \mathcal{Z}_+ \qquad k \in K,\ (i,j) \in E \quad (14)$$

$$z_i^k \in \{0,1\} \qquad k \in K,\ i \in V_h. \quad (15)$$

The objective function (1) takes into consideration the difference between the total routing cost and the total saving in inserting potential customers of level I and II for the current day.

Constraints (2)–(4) mean that each required customer for the current day is serviced exactly once (assignment constraints for required customers). Constraints (5)–(7) ensure that each potential customer for the current day is serviced at most once (assignment constraints for potential customers). Constraints (8) model the demand limitations imposed by the vehicle capacity (knapsack constraints). Constraints (9) model the symmetry conditions at each vertex (flow constraints). Note that, together with the integrality conditions, they also imply parity conditions at each vertex. Constraints (10) are connectivity constraints. They impose that for each subset of vertices excluding the depot and containing a required or potential customer serviced by a vehicle, at least two links incident to the subset must be used to visit it (deadheaded or serviced); they also eliminate subtours disjoint from the depot. Finally, constraints (11)–(15) define the variable domains. We derived the formulation just described from the integer linear programming model proposed in [11] for the mixed capacitated general routing problem.

Note that all customers can be considered required on the last day of a bounded planning horizon since all demands must be fulfilled within the same horizon.

3.3 A First Comparison

Here we present an experiment on a simple instance with the aim of better explaining the two strategies and carrying out a first comparison between them.

Consider the mixed graph depicted in Figure 2. Vertex 1 represents the depot at which three identical vehicles of capacity 15 are based. The sets of elements that must be serviced at least once over a seven-day planning horizon are: $\hat{V} = \{5, 7, 8\}$, $\hat{A} = \{(2, 3), (8, 6), (10, 12), (11, 12)\}$, and $\hat{E} = \{(3, 4), (3, 5), (9, 10)\}$. The service requests were randomly generated by considering the initial outstanding work to be zero. For each vertex in $\hat{V}$ and for each day of the planning horizon, we generated a demand equal to zero with a 10% probability, equal to four with a 20% probability, equal to seven with a 30% probability, and equal to 10 with a 40% probability. A non-zero demand of a vertex requiring a service was labelled as urgent. For each link in $\hat{A} \cup \hat{E}$ and for each day of the planning horizon, we generated a demand in the set {0,1,2,3} with a probability of 25% associated with each value. In order to fix the priorities, we defined three groups. The links (3,4) and (11,12) were assigned to group 1, the links (2,3) and (9,10) were assigned to group 2, the links (3,5), (8,6), and (10,12) were assigned to group 3. In particular, for group 1 a non-zero demand was labelled as unimportant with a 60% probability, as prominent with a 30% probability, and as urgent with a 10% probability; for group 2 a non-zero demand was labelled as unimportant with a 40% probability, as prominent with a 40% probability, and as urgent with a 20% probability; for group 3 a non-zero demand was labelled as unimportant with a 30% probability, as prominent with a 40% probability, and as urgent with a 30% probability. Table 1 reports pairs (demand - priority) generated as mentioned above. Specifically, if the demand value is positive, then the priority is denoted as "u" if the demand is urgent, as "p" if the demand is prominent, and as "t" if the demand is unimportant. If the demand value is equal to 0, pair (demand - priority) is replaced by 0.

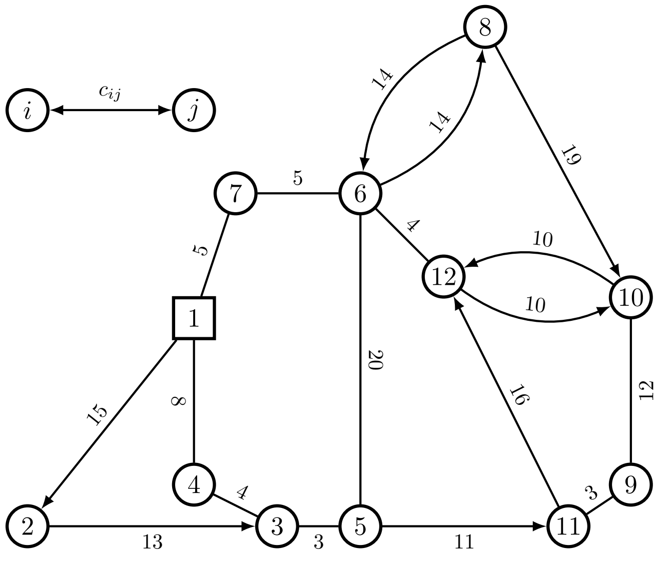

Figure 2: $G = (V, E, A)$.

In order to tackle the DMPGRP by the greedy strategy, we used the branch-and-cut algorithm developed by Bosco et al. [11]. Specifically, seven mixed capacitated general routing problems were solved to optimality. The results are summarized in Table 2. In addition, we modified the above mentioned algorithm to solve the daily problem defined by (1)–(15) within the spread strategy. We set α and β equal to the smallest integer greater than or equal to the average and to the smallest traversal cost, respectively. The results are summarized in Table 3. In this case, pairs (aggregate demand - priority) arise. For day 1, the solution does not include arcs (2,3) and (11,12). The first link represents a potential customer of level I for day 1. Its unserviced demand ($p^1_{2\,3} = 1$) becomes additional to the demand of the next day ($t^2_{2\,3} = 1$) by generating an aggregate demand $d^2_{2\,3}$ equal to 2; moreover, (2,3) becomes a required customer for day 2 since a prominent demand becomes urgent after one day. Therefore, (2 - u) arises for day 2 in correspondence with arc (2,3) in Table 3. Arc (11,12) represents a potential customer of level II for day 1. Its unserviced demand ($p^1_{11\,12} = 1$) becomes additional to the demand of the next day (equal to 0) by generating an aggregate demand $d^2_{11\,12}$ equal to 1; moreover, (11,12) remains

Table 1: Arrivals over the planning horizon

Customers	**Days**						
	1	**2**	**3**	**4**	**5**	**6**	**7**
5	10 - u	0	4 - u	4 - u	7 - u	4 - u	10 - u
7	10 - u	10 - u	10 - u	4 - u	10 - u	0	10 - u
8	7 - u	10 - u	0	10 - u	7 - u	10 - u	7 - u
(2,3)	1 - p	1 - t	2 - p	2 - t	3 - u	0	0
(3,4)	0	2 - t	3 - t	1 - t	2 - p	1 - p	0
(3,5)	1 - u	3 - t	3 - u	1 - t	2 - t	0	1 - p
(8,6)	2 - u	3 - t	2 - p	3 - p	1 - p	3 - p	3 - t
(9,10)	3 - p	3 - t	2 - p	0	0	1 - t	1 - u
(10,12)	1 - u	2 - p	0	2 - t	2 - u	3 - u	0
(11,12)	1 - t	0	2 - t	0	3 - t	1 - p	0

a potential customer of level II for day 2 since unimportant demand does not become prominent after one day. Therefore, (1 - t) arises for day 2 in correspondence of (11,12) in Table 3. We point out that arc (11,12) is also unserviced in day 2. Its demand becomes additional to the demand of the next day ($d^3_{11\,12} = 1 + 2 = 3$); moreover, (11,12) becomes a potential customer of level I for day 3 since unimportant demand concerning day 1 becomes prominent after two days. Therefore, (3 - p) arises for day 3 in correspondence of (11,12) in Table 3. Analogous considerations can be made for the other days of the planning horizon and for the other non-serviced links. On the last day, all priorities associated with non-zero demands are labelled as urgent.

The results in Tables 2 and 3 emphasize that both strategies construct routes which, in seven days, fulfill the total demand equivalent to 217. Anyway, the spread strategy leads to a significant cost saving since the total cost moves from 1,036 to 847 (reduction of 18.24%). The total computational time is about 1.5 seconds for the greedy strategy and 2.5 seconds for the spread strategy.

Table 2: Solution of the greedy strategy

	Days							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**Sum**
Total demand	36	34	28	27	37	23	32	**217**
Total cost	173	147	153	132	160	148	123	**1036**

Table 3: Solution of the spread strategy

	Days							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**Sum**
5	10 - u	0	4 - u	4 - u	7 - u	4 - u	10 - u	
7	10 - u	10 - u	10 - u	4 - u	10 - u	0	10 - u	
8	7 - u	10 - u	0	10 - u	7 - u	10 - u	7 - u	
(2,3)	1 - p	2 - u	2 - p	4 - u	3 - u	0	0	
(3,4)	0	2 - t	3 - t	1 - t	2 - p	1 - p	0	
(3,5)	1 - u	3 - t	6 - u	1 - t	2 - t	0	1 - u	
(8,6)	2 - u	3 - t	2 - p	5 - u	1 - p	3 - p	6 - u	
(9,10)	3 - p	3 - t	5 - p	5 - u	0	1 - t	2 - u	
(10,12)	1 - u	2 - p	2 - u	2 - t	4 - u	3 - u	0	
(11,12)	1 - t	1 - t	3 - p	3 - u	3 - t	4 - p	4 - u	
Total demand	34	27	25	37	36	18	40	**217**
Non-serviced links	(2,3) (11,12)	(3,5) (9,10) (10,12) (11,12)	(2,3) (8,6) (9,10) (11,12)	(10,12)	(11,12)	(8,6) (9,10) (11,12)		
Objective function value (λ)	112	92	72	157	107	86	159	
Real total cost (λ_1)	123	98	75	163	132	97	159	**847**

4 A Heuristic Algorithm for the Spread Strategy

Computing an optimal solution for model (1)–(15) is very time consuming for instances of realistic sizes. In addition, we observe that an optimal solution for the DMPGRP does not consist in juxtaposing optimal daily solutions of model (1)–(15). In effect, new information could reveal failures in past decisions, which cannot be revoked in the future.

For the daily problem of selecting customers and defining vehicle routes within the spread strategy we propose a heuristic algorithm that also uses adaptive large neighbourhood search (ALNS). This solution paradigm was introduced by Ropke and Pisinger [26] and has since been successfully applied to many routing problems in deterministic and stochastic settings [21, 22]. Here we briefly describe the steps carried out to heuristically solve the delivery problem concerning on day h. First, we generate routes servicing only the required customers for day h (*initial partial solution*). Second, we use ALNS to improve the initial partial solution. Third, we try to include in the improved partial solution potential customers of level I and II for day h. This way, we generate a *complete solution* in the sense that we consider all potential and required customers involved in the daily process. We then reapply ALNS to try improving the complete solution. More details on the single steps are given in the following subsections.

In our solution framework, the k^{th} route planned for the current day is represented by an ordered list of elements corresponding to required and potential vertices and links: $\tau_k = [e_1^k, e_2^k, ...]$ (to simplify the notation we do not use the index h). These elements arise in the order in which they are serviced in the route. In addition, the direction in which the edges are traversed during the service is given. For instance, if edge $(i, j) \in E_h$ is serviced by traversing from j to i, it will be present in the list as (j, i). We always assume that a shortest path is followed between the depot and the first serviced element; between two consecutive serviced elements; between the final serviced element and the depot. Moreover, we always construct feasible routes with respect to the capacity constraints. However, the number of routes planned for the current day could exceed $|K|$ during the search. If an *infeasible solution* of this type arises, then we use a *penalty term* in computing the cost of the solution. In particular, this term is given by $\rho \times \max\{0, m - |K|\}$, where ρ is a self-adjusting coefficient and m represents the number of routes in the current solution. Then, for the current day, a solution τ is represented by $\tau = (\tau_1, ..., \tau_m)$.

4.1 Generation of an Initial Partial Solution

We propose a greedy sequential procedure that generates an initial partial solution by constructing one route at a time. In the following, we provide a formal description of our method. Recall that only the required customers for the current day are considered in this phase. For each candidate element to be included in the current route, we define the *distance* as the cost of the shortest path used to connect it to the last element in the current route (to the depot if this route is empty). The procedure is as follows:

Step 0. Set $k = 1$ and $R = V_h^R \cup A_h^R \cup E_h^R$.

Step 1. Initialize route τ_k (it is an ordered and initially empty list).

Step 2. Define $Z \subseteq R$ as the set of required elements that can be added to the end of τ_k without violation of the capacity constraints. If $Z = \emptyset$, then set $k = k + 1$ and go to Step 1.

Step 3. Select the element from Z associated with the smallest distance value (the first element found during the search is selected if more elements are associated with the smallest value). Extend the current route by the selected element and eliminate it from R.

Step 4. If $R = \emptyset$ stop, otherwise go to Step 2.

4.2 Improvement of the Initial Partial Solution

We use an iterative search scheme, named ALNS, with the aim of improving the initial partial solution. In the following, we summarize the main components of an ALNS algorithm by pointing out parameters values, criteria and routines used in our scheme.

First, every ALNS algorithm needs acceptance and stopping criteria, like all iterative methods. In our scheme, we accept feasible or infeasible solutions that are better, equal or slightly worse than the current solution. In other words, a feasible or infeasible solution is accepted if its objective value is less than the objective value of current solution multiplied by a user-defined factor $\delta \geq 1$. Note that we just refer to term λ_1 in (1). In order to accept fewer and fewer worsening solutions during the search, the parameter δ gradually decreases until it becomes equal to 1. Its initial value is $\delta = 1.03$. Every 10 iterations δ is updated in the following way: $\delta = \max\{1, \delta \times 0.999\}$. Similarly, in order to reduce the generation of infeasible solutions during the search, the self-adjusting coefficient ρ can only increase. Its initial value is $\rho = 1$. Every 10 iterations, the coefficient is multiplied by $2^{(b/10)}$, where b denotes the number of infeasible solutions encountered in the last 10 solutions. The algorithm terminates whenever the best feasible solution has not changed for $\gamma = 50$ consecutive iterations.

In addition, every ALNS algorithm needs a set of *destroy* and *repair* operators. At each iteration of the ALNS, one destroy operator and one repair operator are applied to the current solution. The destroy operator removes q elements in R from the current solution and the repair operator reinserts them. We recall that $R = V_h^R \cup A_h^R \cup E_h^R$. In our implementation, we set $q = \lceil 0.20|R| \rceil$. The selection of destroy and repair operators is based on a roulette-wheel selection mechanism. In other words, the destroy and repair operators are associated with specific *weights*. Given t operators with weights w_i, the j^{th} one is selected with probability $w_j / \sum_{i=1}^{t} w_i$. Note that the destroy operator is selected independently of the repair operator (and vice versa). The weights of the operators change during the search according to their effectiveness. Specifically, the ALNS is divided into a number of *segments* of ϱ iterations. We used $\varrho = 30$ in our implementation. The weights are updated every ϱ iterations by using the scores obtained during the last segment. In the first segment the weight of every operator is equal to 1, and at the start of a segment the score of every operator is equal to 0. The score of the selected pair of destroy and repair operators is increased by 30, 10, and 5 if their application results, respectively, in a new best feasible solution, in a (possibly infeasible) solution improving the current one, in an accepted (possibly infeasible) solution not improving the current one. At the end of every segment, new weights are calculated in the following way. Let $w_{i,j}$, $\pi_{i,j}$, and $\theta_{i,j}$ be the weight of the i^{th} operator in the j^{th} segment, the score of the i^{th} operator obtained during the j^{th} segment, and the number of times the i^{th} operator has been used during the j^{th} segment, respectively. If $\theta_{i,j} = 0$, then set $w_{i,j+1} = w_{i,j}$; otherwise, set $w_{i,j+1} = 0.9w_{i,j} + 0.1(\pi_{i,j}/\theta_{i,j})$. We now present the destroy and the repair operators included in our heuristic scheme.

4.2.1 Destroy Operators

Three operators are used to destroy the current solution, i.e., remove q customers from the routes where they are serviced.

The first operator is called *Worst Removal* (WR). It selects customers that appear to be placed in a wrong position in the current solution. Specifically, WR computes the saving obtained by removing every customer from the route in which it is serviced. The operator repeatedly chooses the customer associated with the largest saving until q customers have been removed.

The second operator is called *Random Removal* (RR). This operator selects (almost all) customers at random with the aim of diversifying the search. If the removal of a customer leads to a null saving, then the customer serviced immediately after in the same route is also removed (or the customer serviced immediately before if no customer is serviced after). Note that the removal of a customer belonging to a shortest path between the customer serviced immediately before (or the depot if no customer is serviced before) and the customer serviced immediately after (or the depot if no customer is serviced after) is associated with a null saving. Of course, in the case of a null saving, RR only removes those customers serviced immediately after (or before) if the limit of q removals has not yet been exceeded.

The third operator is called *Route Deletion* (RD). It tries to empty routes. In particular, RD tries to regain feasibility whenever the number of routes in the current solution is larger than the number of available vehicles. Operatively, the routes are sorted in non-decreasing order of the number of customers serviced by them. Then, the operator considers the first route and iteratively removes its customers starting from the last one. Moreover, it considers the next route if the previous one has been emptied.

4.2.2 Repair Operators

Three operators are used to repair the current solution, i.e., reinserting the q customers removed in the previous step (and stored in a list χ). We recall that only infeasibility with respect to the number of routes is allowed during the search. Therefore, the operators never consider insertions that violate the capacity constraints. In addition, they evaluate the insertion of an edge in both directions.

The first repair operator is called *Greedy Construction* (GC). At each iteration, the operator extracts from χ the customer associated with the minimum insertion cost. It then inserts this customer at its cheapest position. The process continues until all customers have been inserted. Note that GC considers the insertion in all existing routes (i.e., all routes including one customer at least) and, in addition, in a new empty route.

The second operator is called *Best Postponement* (BP). For every customer in χ, it computes the insertion cost in the cheapest position. Then, BP extracts and inserts in this position the customer whose insertion cost is maximum. The underlying idea is to avoid even higher costs in next iterations.

The third operator uses look-ahead information when selecting the customer to insert. It is called *Regret Maximization* (RM). For each customer in χ, the operator computes a *regret value* as the difference in cost between inserting the customer in the cheapest position in its second-best route and in its best route. An insertion in a new route is possible. If the best route is a new route, then the second-best route can be a new route only in absence of existing routes consistent with the capacity constraints (in this case the regret value is equal to 0). RM selects from χ the customer for which the regret value is maximum and inserts it in its best position.

4.3 Insertions for Defining a Complete Solution

In order to generate a complete solution, we insert potential customers for the current day in the improved partial solution by means of a two-phase procedure. In the first phase, we consider zero-cost insertions. In other words, potential customers of level I and II for the current day already present in some existing route (since belonging to a shortest path) pass from the state "unserviced" to state "serviced" if the operation is consistent with the capacity constraints. In the second phase, an attempt for other insertions is made. In particular, potential customers of level I for the current day will be included in existing routes if the insertion cost does not exceed α, whereas potential customers of level II for the current day will be included in existing routes if the insertion cost does not exceed β. Note that these operations are fast, since the potential customers are inserted in the first position for which the condition is satisfied. In both phases, potential customers of level I have precedence over potential customers of level II since their "deadline" is closer. The ALNS algorithm described in the previous subsection is then reapplied with the aim of improving the overall solution. In this phase, the set R defined in Section 4.1 is replaced with the set of all required and potential customers serviced in the complete solution. The final part of the algorithm (definition and improving of a complete solution) is not applied on the last day of a bounded planning horizon since all customers are required on that day.

5 Computational Experiments

Computational experiments were carried out on a PC equipped with an Intel Core i7 CPU running at 2.40 GHz, with 16 GB of memory. The heuristic algorithm used within the spread strategy and described in Section 4 was coded in C++. The computational results associated with the greedy strategy were obtained by running the exact algorithm coded in Java by Bosco et al. [11]. In particular, for each day of the planning horizon, the branch-and-cut algorithm of Bosco et al. was used to solve a mixed capacitated general routing problem to optimality, as shown in Section 3.3. Finally, a modified version of this branch-and-cut algorithm was used to optimally solve model (1)–(15) with the aim of obtaining values of comparison for the heuristic algorithm described in Section 4. More specifically, in a preliminary experimental phase (tuning phase) we solved to optimality model (1)–(15) with different values of α and β on a set of instances of very small size like the one described in Section 3.3. The numerical results led us to set values for α and β equal to the smallest integer greater than or equal to the average and to the smallest traversal cost, respectively. Recall that the parameters α and β also play an important role in the heuristic algorithm. Indeed, these are not used in evaluating the objective value of the solutions generated during the search, but are used in defining the complete solution and, consequently, in selecting potential customers on a daily basis (see Section 4.3). We also used the results of the tuning phase to make decisions about operators and parameters in the ALNS heuristic.

5.1 Instances

We then carried out an intensive experimental phase based on instances derived from dataset mggdb-0.25 having $|K| \leq 7$ vehicles. This dataset was proposed by Bosco et al. [11] and is available on the Transportation Optimization Portal of SINTEF Applied Mathematics [27]. Bosco et al. derived mggdb-0.25 instances from gdb instances introduced for the undirected capacitated arc routing problem [18]. Specifically, the authors modified the original gdb instances in the following way. First, in order to switch from undirected to mixed graphs, they replaced a certain number of edges with pairs of opposite arcs and moved the demand of each required edge to one (randomly chosen) of the two arcs. Second, for dataset mggdb-0.25, they shifted the demands of $\lceil 0.25\ell \rceil$ randomly selected required links to $\lceil 0.25\ell \rceil$ randomly selected adjacent vertices, where ℓ is the number of links requiring service in their mixed graphs. In dataset mggdb-0.25, the number of vertices, arcs, and edges varies from seven to 22, from 18 to 68, and from two to 11, respectively.

From this dataset we saved (i) the overall structure of the mixed graphs; (ii) the traversal costs for every link; (iii) the number of the homogeneous vehicles.

The demands were randomly generated over a seven-day planning horizon by considering specific rules. We used the set of vertices and links requiring service in the mggdb-0.25 dataset to define the elements of $\hat{V} \cup \hat{A} \cup \hat{E}$ and their demands as follows. For each vertex requiring a service in the original instance and for each day of the planning horizon, we generated a demand equal to zero with a 10% probability, equal to four with a 20% probability, equal to seven with a 30% probability, and equal to 10 with a 40% probability. A non-zero demand of a vertex requiring a service was always labelled as urgent, as done for the instance described in Section 3.3. For each link requiring a service in the original instance and for each day of the planning horizon, we generated a demand in set {0,1,2,3} with a probability of 25% associated with each value. In order to fix the priorities, we defined three groups. Each link requiring a service was randomly assigned to a group with the same probability of assignment for each group. The probabilities of generating urgent, prominent or unimportant demands associated with each group are equal to those described in Section 3.3.

The capacity of the vehicles was modified with respect to the original one, coherently with the new demand values. The new dataset, including 19 instances, is denoted as dmpgrp-0.25. In particular, dmpgrp-0.25-1 represents the instance derived from mggdb-0.25-1, dmpgrp-0.25-2 represents the instance derived from mggdb-0.25-2, and so on.

5.2 Main Numerical Results

In our final experimental phase, we ran our heuristic algorithm five times for each instance by using five different random seeds. Since the planning horizon is bounded, the objective value to be minimized corresponds to the total routing cost over the planning horizon.

In Table 4, the columns "Greedy-Exact" report the results obtained by applying the greedy strategy. "Exact" refers to the fact that we used an exact algorithm to compute the solutions. The columns "Spread-Heuristic" report the results obtained by the spread strategy through the heuristic described in Section 4. The other column headings in Table 4 are defined as follows:

$Name$	instance name
$\vert K\vert$	number of vehicles
Q	vehicle capacity
η	number of elements (vertices and links) requiring service
Dem	total demand serviced over the planning horizon
$Cost^*$	optimal total routing cost of the solution for "Greedy-Exact"
$Cost_A$	average total routing cost of the solution for "Spread-Heuristic"
$Cost_B$	best total routing cost of the solution for "Spread-Heuristic"
Sec	computation time in seconds
Imp	percentage improvement computed as $100(Cost^* - Cost_B)/Cost^*$.

Since in any feasible solution for the DMPGRP all demands are fulfilled within the planning horizon, the total demand Dem is the same for "Greedy-Exact" and "Spread-Heuristic".

Note that $Cost^*$, $Cost_A$, and $Cost_B$ correspond to the total routing cost, i.e. to the real cost of the solutions. Recall that for each day of the planning horizon, the routing cost under the spread strategy is defined by λ_1 in (1). Consequently, a fair comparison between "Greedy-Exact" and "Spread-Heuristic" is ensured in terms of solution quality. The computation time is the sum of the times to optimally solve the mixed capacitated general routing problem for each day of the planning horizon in columns "Greedy-Exact", but it is the total time for the five runs in columns "Spread-Heuristic". Table 4 shows that the spread strategy leads to significant improvements with respect to "Greedy-Exact". In particular, Imp varies between 14.62% and 35.28%, with an average value equal to 27.19%. Such a significant improvement is impressive for several reasons:

- For the DMPGRP solved through the greedy strategy, we always reached an optimal solution value. For the DMPGRP solved through the spread strategy we only report the value of the best feasible solution obtained through a heuristic algorithm.
- The spread strategy does not eliminate the issue of "forced" deliveries that characterize the greedy strategy; indeed, mandatory deliveries of initially prominent or unimportant items are made on the days in which they become urgent.
- The "forced" deliveries that we impose on the last day of the planning horizon in order to ensure the condition "all demands must be fulfilled within the planning horizon" are very heavy for the spread strategy, especially because the size of the planning horizon is small in our computational experiments (seven days).

In real contexts, where the planning horizon is almost unbounded and the "forced" deliveries are generally rarer than those imposed within the spread strategy, the improvements obtained by this solution framework may be even more significant.

We stress that the heuristic algorithm described in Section 4 is very fast. Indeed, the computational times are less than one second for all the instances except one for which a little more than three seconds is required. In contrast, the search of the optimal solution within the greedy strategy requires much larger times (more than 69 hours for instance dmpgrp-0.25-12), which may be unacceptable in real-world applications. Obviously, the branch-and-cut algorithm used in the greedy strategy may be replaced with an efficient (meta)heuristic by accelerating its speed significantly without sacrificing solution quality very much (e.g., the solution method proposed by Vidal [30]). However, here we compare

Table 4: Computational results for dataset dmpgrp-0.25

					Greedy-Exact		**Spread-Heuristic**			
Name	$\|K\|$	Q	η	*Dem*	$Cost^*$	*Sec*	$Cost_A$	$Cost_B$	*Sec*	*Imp*
dmpgrp-0.25-1	5	20	21	447	1457	558.4	1203.2	1191	0.6	18.26
dmpgrp-0.25-2	6	18	25	518	1920	48227.4	1465.4	1429	0.7	25.57
dmpgrp-0.25-3	5	20	22	502	1595	1696.2	1277.2	1251	0.5	21.57
dmpgrp-0.25-4	4	19	18	383	1790	53.0	1374.6	1351	0.4	24.53
dmpgrp-0.25-5	6	17	24	445	2118	10491.4	1589.6	1566	0.5	26.06
dmpgrp-0.25-6	5	19	21	437	1711	2409.6	1242.6	1222	0.5	28.58
dmpgrp-0.25-7	5	17	20	405	1760	803.5	1438.2	1410	0.4	19.89
dmpgrp-0.25-10	4	21	22	411	1669	33.7	1109.8	1083	0.7	35.11
dmpgrp-0.25-11	5	28	41	702	1894	111491.7	1461.4	1404	3.4	25.87
dmpgrp-0.25-12	7	12	22	399	2879	248911.8	2483.0	2458	0.4	14.62
dmpgrp-0.25-13	6	15	26	448	2105	11432.7	1494.8	1467	0.6	30.31
dmpgrp-0.25-14	5	16	20	424	609	9.4	399.6	396	0.4	34.98
dmpgrp-0.25-15	4	21	20	444	343	6.2	226.0	222	0.4	35.28
dmpgrp-0.25-16	5	16	25	442	511	2475.8	392.6	388	0.6	24.07
dmpgrp-0.25-17	5	16	25	475	373	2.1	289.0	281	0.8	24.66
dmpgrp-0.25-18	5	17	32	456	700	172618.5	542.0	529	0.9	24.43
dmpgrp-0.25-19	3	16	10	233	321	0.9	208.0	208	0.1	35.20
dmpgrp-0.25-20	4	18	20	382	698	2.1	463.2	455	0.5	34.81
dmpgrp-0.25-21	6	19	31	536	741	13153.8	507.8	498	0.9	32.79
									Average	**27.19**

the results of the spread strategy with the optimal ones provided by the competitor in order to emphasize the potential gap in terms of solution quality when delivery flexibility is introduced in planning.

5.3 Heuristic: Effect of the Single Operators

The structure of the heuristic is light in order to obtain short solution times for real-world cases. Through the numerical results reported in Tables 5 and 6, we analyse how the operators described in Sections 4.2.1 and 4.2.2 contribute to the improvement in terms of solution quality. Specifically, Tables 5 and 6 show the effect of eliminating one or more operators used to repair or to destroy the current solution, respectively.

We have carried out a comparison of the percentage improvements, computed as in

Table 4, by referring to the cost of the best solution over five runs and to $Cost^*$, i.e., to the total cost of the solution found by solving to optimality a mixed capacitated general routing problem for each day of the planning horizon. In particular, the columns "All" refer to the version of the algorithm in which all operators are used (therefore, its values coincide with the values reported in the final column of Table 4). The other columns refer to the version of the algorithm obtained by using a restricted number of operators. In particular, in Table 5, the column headings show the operators kept to repair the current solution. Analogously, in Table 6, the column headings show the operators kept to destroy the current solution. For instance, column "GC" refers to the version of the algorithm obtained by using only GC to repair the current solution, i.e., eliminating the repair operators BP and RM (Table 5), whereas column "RR+RD" refers to the version of the algorithm obtained by using only the destroy operators RR and RD, i.e., eliminating the destroy operator WR (Table 6). Note that we never eliminated the destroy operator RR, since randomness is fundamental in our algorithm in order to ensure a search diversification.

The numerical results reported in Table 5 show that the repair operators GC, BP, and RM contribute to increasing the quality of the solution. In particular, RM seems to provide a better contribution than those of the other two operators. This observation is supported by the average percentage improvement obtained both when it is eliminated (24.98% is the smallest average value in columns "BP+RM", "GC+RM","GC+BP") and when it is used to repair the current solution alone (24.43% is the largest average value in columns "GC", "BP","RM"). Between GC and BP, GC seems to be superior for the same reasons.

The numerical results reported in Table 6 show that the decrease of the average percentage improvement associated with the elimination of WR is lightly larger than the decrease of the average percentage improvement associated with the elimination of RD. The effect of their elimination is not dissimilar from the effect of the elimination of the operators used to repair the current solution.

In conclusion, as shown in Tables 5 and 6, all the operators used to destroy and repair the solutions during the search contribute to the solution quality. More specifically, we never registered a higher value of Imp through the elimination of any operator.

Table 5: Percentage improvements to evaluate the repairing effect

Name	All	BP+RM	GC+RM	GC+BP	GC	BP	RM
dmpgrp-0.25-1	18.26	17.23	16.61	16.27	15.17	16.06	16.20
dmpgrp-0.25-2	25.57	22.97	23.23	22.71	22.34	22.60	22.66
dmpgrp-0.25-3	21.57	18.24	17.18	17.49	16.18	16.93	15.86
dmpgrp-0.25-4	24.53	21.01	22.40	21.28	21.01	20.00	21.01
dmpgrp-0.25-5	26.06	24.60	24.55	24.41	22.43	23.89	24.13
dmpgrp-0.25-6	28.58	26.77	26.77	27.24	26.07	26.71	25.66
dmpgrp-0.25-7	19.89	17.27	17.56	17.78	17.27	16.88	16.76
dmpgrp-0.25-10	35.11	32.95	32.95	32.95	32.12	29.90	32.95
dmpgrp-0.25-11	25.87	20.80	20.33	22.18	19.38	19.64	20.12
dmpgrp-0.25-12	14.62	10.91	11.22	11.98	10.91	9.59	10.84
dmpgrp-0.25-13	30.31	28.98	28.98	28.41	27.74	28.36	28.27
dmpgrp-0.25-14	34.98	34.81	34.81	34.65	33.33	33.17	34.65
dmpgrp-0.25-15	35.28	33.53	33.53	32.94	32.36	31.78	32.36
dmpgrp-0.25-16	24.07	22.90	22.50	21.53	20.35	19.57	21.33
dmpgrp-0.25-17	24.66	21.18	23.06	21.18	20.91	18.77	20.64
dmpgrp-0.25-18	24.43	22.71	22.43	22.14	21.43	21.29	22.14
dmpgrp-0.25-19	35.20	35.20	35.20	35.20	35.20	35.20	35.20
dmpgrp-0.25-20	34.81	33.52	33.38	33.24	32.95	32.23	33.09
dmpgrp-0.25-21	32.79	30.77	31.17	31.04	30.23	28.34	30.36
Average	**27.19**	**25.07**	**25.15**	**24.98**	**24.07**	**23.73**	**24.43**

Table 6: Percentage improvements to evaluate the destroying effect

Name	All	RR+RD	RR+WR	RR
dmpgrp-0.25-1	18.26	15.31	15.51	14.07
dmpgrp-0.25-2	25.57	23.65	22.71	22.71
dmpgrp-0.25-3	21.57	17.81	19.06	16.87
dmpgrp-0.25-4	24.53	20.34	19.83	19.44
dmpgrp-0.25-5	26.06	23.04	24.83	22.80
dmpgrp-0.25-6	28.58	28.23	28.00	28.00
dmpgrp-0.25-7	19.89	17.90	17.39	17.39
dmpgrp-0.25-10	35.11	33.49	33.49	33.19
dmpgrp-0.25-11	25.87	21.07	22.70	19.85
dmpgrp-0.25-12	14.62	11.60	11.77	11.57
dmpgrp-0.25-13	30.31	29.03	28.88	28.03
dmpgrp-0.25-14	34.98	34.48	34.32	34.15
dmpgrp-0.25-15	35.28	33.53	33.53	33.53
dmpgrp-0.25-16	24.07	20.55	22.70	20.35
dmpgrp-0.25-17	24.66	23.06	21.98	21.72
dmpgrp-0.25-18	24.43	22.00	21.71	21.29
dmpgrp-0.25-19	35.20	35.20	35.20	35.20
dmpgrp-0.25-20	34.81	33.95	33.95	33.81
dmpgrp-0.25-21	32.79	30.36	30.63	30.36
Average	**27.19**	**24.98**	**25.17**	**24.44**

Table 7: Computational results for two variants of dmpgrp-0.25-1

	Greedy-Exact		**Spread-Heuristic**		
Name	*Cost**	*Sec*	*Cost*$_B$	*Sec*	*Imp*
dmpgrp-0.25-1-var1	1520	347.6	1096	0.6	27.89
dmpgrp-0.25-1-var2	1563	416.5	1259	0.6	19.45

5.4 Further Experiments

Further experiments were carried out in order to get more information from both the problem and the heuristic behaviour.

First, we regenerated the demands over the seven-day planning horizon for the elements requiring a service in the dmpgrp-0.25-1 instance (we kept the graph structure, the elements requiring a service, and the values of the other parameters). Specifically, we generated two variants for this instance naming them dmpgrp-0.25-1-var1 and dmpgrp-0.25-1-var2, respectively. Variant dmpgrp-0.25-1-var1 is associated with a total demand equal to 436, whereas variant dmpgrp-0.25-1-var2 is associated with a total demand equal to 435. In summary, the dmpgrp-0.25-1 instance and its variants have a very similar structure and computational complexity. Their total demand is also similar. We ran the ALNS algorithm five times with five random seeds to find a solution (the best one) for the spread strategy. In addition, we ran the branch-and-cut algorithm of Bosco et al. to find a solution for the greedy strategy. The results for dmpgrp-0.25-1-var1 and dmpgrp-0.25-1-var2 are summarized in Table 7, where the columns "Greedy-Exact" refer to the results obtained by the greedy strategy and the columns "Spread-Heuristic" refer to the results obtained by the spread strategy through the ALNS algorithm. The other column headings in Table 7 (i.e., *Name*, *Cost**, *Cost*$_B$, *Sec* and *Imp*) are defined as in Table 4. We note that different demand distributions over the planning horizon led to quite different percentage improvements, especially for the first variant.

For the dmpgrp-0.25-1 instance we also increased the number of urgent requests. Specifically, we transformed this instance by changing the labels of some prominent and unimportant requests from non-urgent to urgent, and keeping the value of all the demands. We still ran the ALNS algorithm five times with five random seeds and selected the best total routing cost. Note that this transformation did not change the solution for the greedy strategy. When the 25% of prominent and unimportant requests became urgent (through a random choice), we obtained a percentage improvement equal to 14.69%, when the 50% of prominent and unimportant requests became urgent, we obtained a percentage improvement equal to 8.79%, when the 75% of prominent and unimportant requests became urgent, we obtained a percentage improvement equal to 3.02%. In the latter case, the urgent requests corresponded to more than 86% of the total requests. We repeated the same experiment with the dmpgrp-0.25-21 instance for which we obtained an Imp value very different from the one associated with the dmpgrp-0.25-1 instance. For the instance dmpgrp-0.25-21, when the 25% of prominent and unimportant requests became urgent, we obtained a 28.07% improvement; when the 50% of prominent and unimportant requests became urgent, we obtained a 19.30% improvement; when the 75% of prominent and unimportant requests became urgent, we obtained a 12.55% improvement. In this last case, the urgent requests corresponded to more than 84% of the total requests and such a large percentage improvement was impressive.

However, the results concerning the last typology of experiments confirmed an expected result. When the percentage of urgent requests increases the percentage improvement decreases. In the extreme case, when all the non-zero demands are labelled as urgent, the DMPGRP "degenerates" into a set of seven mixed capacitated general routing problems. We used this extreme case to evaluate the performance of the ALNS algorithm. Specifically, we ran again the ALNS algorithm after appropriately changing the labels of the non-zero demands: all requests became urgent. The costs of the solutions provided by the ALNS algorithm were compared to those associated with the option "Greedy-Exact" in Table 4. In effect, in the extreme case, the greedy strategy becomes an exact solver for the DMPGRP. Table 8 summarizes the results of the comparison for all the instances. In this table, the column headings "$Name$", "Dem", "$Cost^*$", and "$Cost_B$" have the same meaning as in Table 4. These columns also report the same values of Table 4, except for $Cost_B$ (it still is the best total routing cost over five different runs, but cannot be better than $Cost^*$ in the extreme case). The new column heading "*Gap*" is defined as follows: $100(Cost_B - Cost^*)/Cost^*$.

The results reported in Table 8 confirm that our ALNS algorithm works well, although the solution framework is light and its operators are simple. In effect, the average *Gap* is equal to 1.82%. In addition, the maximum value of *Gap* is less than an acceptable threshold of 5%. No optimal solution was obtained by the heuristic algorithm, but it is worth noting

Table 8: Computational results for the case "only urgent demands"

		Greedy-Exact	**Spread-Heuristic**	
Name	*Dem*	$Cost^*$	$Cost_B$	*Gap*
dmpgrp-0.25-1	447	1457	1525	4.67
dmpgrp-0.25-2	518	1920	1956	1.88
dmpgrp-0.25-3	502	1595	1652	3.57
dmpgrp-0.25-4	383	1790	1833	2.40
dmpgrp-0.25-5	445	2118	2189	3.35
dmpgrp-0.25-6	437	1711	1714	0.18
dmpgrp-0.25-7	405	1760	1763	0.17
dmpgrp-0.25-10	411	1669	1689	1.20
dmpgrp-0.25-11	702	1894	1986	4.86
dmpgrp-0.25-12	399	2879	2914	1.22
dmpgrp-0.25-13	448	2105	2125	0.95
dmpgrp-0.25-14	424	609	610	0.16
dmpgrp-0.25-15	444	343	347	1.17
dmpgrp-0.25-16	442	511	521	1.96
dmpgrp-0.25-17	475	373	377	1.07
dmpgrp-0.25-18	456	700	707	1.00
dmpgrp-0.25-19	233	321	333	3.74
dmpgrp-0.25-20	382	698	699	0.14
dmpgrp-0.25-21	536	741	747	0.81
			Average	**1.82**

that to find an optimal solution for a single instance, seven mixed capacitated general routing problems must be solved to optimality (corresponding to the several days of the planning horizon). Consider, for example, instances dmpgrp-0.25-14 and dmpgrp-0.25-20 for which our ALNS algorithm obtained a solution cost that differs from the optimal one by just one unit. In both cases, the ALNS algorithm optimally solved the sub-instances corresponding to six days. Specifically, for the dmpgrp-0.25-14 instance it did not find the optimal value associated with the fifth sub-instance (day 5); for the dmpgrp-0.25-20 instance it did not find the optimal value associated with the first sub-instance (day 1).

6 Conclusions

We have proposed a dynamic multi-period general routing problem (DMPGRP) arising in postal services and parcel delivery organizations. In our setting, the items to be delivered are gradually revealed over a planning horizon consisting of several days. The decision maker must plan the vehicle routes on a daily basis with the aim of minimizing the operational costs. Two solution strategies were presented. The first one, called greedy strategy, is based on a conventional solution framework proposed in the scientific literature. The second one, called spread strategy, deals with service priorities. It is based on the use of an adaptive large neighbourhood search (ALNS) metaheuristic. Although the DMPGRP is a multi-period problem, this algorithm cannot consider the routes of different periods at the same time, since the data concerning the service demands are revealed over time. For instance, the heuristic algorithm cannot move elements that have to be serviced from a route to another route if they relate to different periods of the planning horizon. The algorithm defines the vehicle routes day by day and cannot modify them when new information becomes available. Computational results have shown that the spread strategy can lead to significant savings with respect to the greedy strategy.

From a practical prospective, our results emphasize the need to investigate aspects of service differentiation and develop decision support instruments that deal with them, like our spread strategy. In addition, we have presented a solution methodology dealing with unpredictable demands that can arise, e.g., in contexts where e-commerce is involved.

We have designed our model and our solution strategies with specific delivery systems in mind, like modern national postal service organizations and private freight carriers. However, they can be adapted to other real environments and situations. In all cases, we considered items to be delivered in contexts of city logistics (small packages shipping) where a relatively small total vehicle capacity is sufficient. In a highly dynamic setting, infeasibility due to an insufficient capacity may arise occasionally. According to Dayarian and Savelsbergh [16], in several delivery systems, due to the uncertainty associated with the arrivals and the limited size of the fleet of company vehicles, it is impossible to guarantee that service is met for all the requests within the desired times. In the case of frequent "failures" (and, consequently, of generalized lateness), the delivery organization may have to increase the size of its vehicle fleet.

Acknowledgements

The work by Gilbert Laporte was partly supported by the Canadian Natural Sciences and Engineering Research Council under grant 2015–06189. This support is gratefully acknowledged. Thanks are due to the Associate Editor and to the reviewers for their valuable comments.

References

[1] Albareda-Sambola M., Fernández E., and Laporte G. (2014). The dynamic multiperiod vehicle routing problem with probabilistic information, Computers & Operations Research, 48: 31–39.

[2] Angelelli E., Bianchessi N., Mansini R., and Speranza M.G. (2009). Short term strategies for a dynamic multi-period routing problem, Transportation Research Part C: Emerging Technologies, 17(2): 106–119.

[3] Angelelli E., Speranza M.G., and Savelsbergh M.W.P. (2007). Competitive analysis for dynamic multiperiod uncapacitated routing problems, Networks, 49(4): 308–317.

[4] Archetti C., Bertazzi L., Laganà D., and Vocaturo F. (2017). The undirected capacitated general routing problem with profits, European Journal of Operational Research, 257(3): 822–833.

[5] Archetti C., Jabali O., and Speranza M.G. (2015). Multi-period vehicle routing problem with due dates, Computers & Operations Research, 61: 122–134.

[6] Azi N., Gendreau M., and Potvin J.-Y. (2012). A dynamic vehicle routing problem with multiple delivery routes, Annals of Operations Research, 199(1): 103–112.

[7] Bach L., Lysgaard J., and Wøhlk S. (2016). A branch-and-cut-and-price algorithm for the mixed capacitated general routing problem, Networks, 68(3): 161–184.

[8] Ben-Tal A. and Nemirovski A. (2002). Robust optimization – methodology and applications, Mathematical Programming, 92(3): 453–480.

[9] Benavent E., Corberán Á., Laganà D., and Vocaturo F. (2019). The periodic rural postman problem with irregular services on mixed graphs, European Journal of Operational Research, 276(3): 826–839.

[10] Berbeglia G., Cordeau J.-F., and Laporte G. (2010). Dynamic pickup and delivery problems, European Journal of Operational Research, 202(1): 8–15.

[11] Bosco A., Laganà D., Musmanno R., and Vocaturo F. (2013). Modeling and solving the mixed capacitated general routing problem, Optimization Letters, 7(7): 1451–1469.

[12] Bosco A., Laganà D., Musmanno R., and Vocaturo F. (2014). A matheuristic algorithm for the mixed capacitated general routing problem, Networks, 64(4): 262–281.

[13] Campbell A.M. and Wilson J.H. (2014). Forty years of periodic vehicle routing, Networks, 63(1): 2–15.

[14] Chen A.I. (2017). Large-scale optimization in online-retail inventory management. Ph.D. Thesis, Massachusetts Institute of Technology, Massachusetts.

[15] Ciancio C., Laganà D., and Vocaturo F. (2018). Branch-price-and-cut for the mixed capacitated general routing problem with time windows, European Journal of Operational Research, 267(1): 187–199.

[16] Dayarian I. and Savelsbergh M.W.P. (2020). Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders, Production and Operations Management, 29(9): 2153–2174.

[17] Florio A.M., Feillet D., and Hartl R.F. (2018). The delivery problem: Optimizing hit rates in e-commerce deliveries, Transportation Research Part B: Methodological, 117(A): 455–472.

[18] Golden B.L., Dearmon J.S., and Baker E.K. (1983), Computational experiments with algorithms for a class of routing problems, Computers & Operations Research, 10(1): 47–59.

[19] Irnich S., Laganà D., Schlebusch C., and Vocaturo F. (2015). Two-phase branch-and-cut for the mixed capacitated general routing problem, European Journal of Operational Research, 243(1): 17–29.

[20] Kovacs A.A., Golden B.L., Hartl R.F., and Parragh S.N. (2014). Vehicle routing problems in which consistency considerations are important: A survey, Networks, 64(3): 192–213.

[21] Laporte G., Musmanno R., and Vocaturo F. (2010). An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands, Transportation Science, 44(1): 125–135.

[22] Liu R., Tao Y., and Xie X. (2019). An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits, Computers & Operations Research, 101: 250–262.

[23] Mourão M.C. and Pinto L.S. (2017). An updated annotated bibliography on arc routing problems, Networks, 70(3), 144–194.

[24] Pérez Rivera A.E. and Mes M.R.K. (2017). Anticipatory freight selection in intermodal long-haul round-trips, Transportation Research Part E: Logistics and Transportation Review, 105: 176–194.

[25] Poste Italiane. Mail, parcel and distribution – Delivery services are an integral part of the history of Poste Italiane, https://www.posteitaliane.it/en/mail-parcel-and-distribution.html. Accessed: 3 April, 2019.

[26] Ropke S. and Pisinger D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, Transportation Science, 40(4): 455–472.

[27] SINTEF Applied Mathematics. Transportation Optimization Portal (TOP) - NEARP/MCGRP, https://www.sintef.no/projectweb/top/nearp. Accessed: 16 April, 2019.

[28] Ulmer M.W., Soeffker N., and Mattfeld D.C. (2018). Value function approximation for dynamic multi-period vehicle routing, European Journal of Operational Research, 269(3): 883–899.

[29] van Heeswijk W.J.A., Mes M.R.K., and Schutten J.M.J. (2019). The delivery dispatching problem with time windows for urban consolidation centers, Transportation Science, 53(1): 203–221.

[30] Vidal T. (2017). Node, edge, arc routing and turn penalties: Multiple problems–one neighborhood extension, Operations Research, 65(4): 992–1010.

[31] Wen M., Cordeau J.-F., Laporte G., and Larsen J. (2010). The dynamic multi-period vehicle routing problem, Computers & Operations Research, 37(9): 1615–1623.

[32] Zhang S., Ohlmann J.W., and Thomas B.W. (2018). Dynamic orienteering on a network of queues, Transportation Science, 52(3): 691–706.