A Practical Time Slot Management and Routing Problem for Attended Home Services

Bruno P. Bruck[a,*], Jean-François Cordeau[b], Manuel Iori[a]

[a] *Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Via Amendola 2, 42122 Reggio Emilia, Italy*
[b] *HEC Montréal and CIRRELT, 3000, chemin de la Côte-Sainte-Catherine, Montréal, Canada, H3T 2A7*

Abstract

This paper describes the solution methodology developed to address an attended home delivery problem faced by an Italian provider of gas, electricity, and water services. This company operates in several regions and must dispatch technicians to customer locations where they carry out installation or maintenance activities within time intervals chosen by the customers. The problem consists of creating time slot tables specifying the amount of resources allocated to each region in each time slot, and of routing technicians in a cost-effective way. We propose a large neighborhood search (LNS) heuristic to create time slot tables by relying on various simulation strategies to represent the behavior of customers and on an integer linear program to optimize the routing of technicians. In addition, we also use a second integer program as a repair mechanism inside the LNS heuristic. Computational experiments carried out on data provided by the company confirm the efficiency of the proposed methodology.

Keywords: Attended home delivery, time slot management, routing, large neighborhood search, integer linear programming

*Corresponding author
Email addresses: bruno.petratobruck@unimore.it (Bruno P. Bruck), jean-francois.cordeau@hec.ca (Jean-François Cordeau), manuel.iori@unimore.it (Manuel Iori)

1. Introduction

Attended home delivery (AHD) is a last-mile service that requires the attendance of customers and it is an important distribution policy adopted in many industries, especially in e-grocery, where profit margins are typically low. The design of efficient and reliable delivery plans is a critical aspect for the success in this market. In addition, in order to meet customer needs in the best possible way, a certain quality of service (QoS) level must be provided while balancing delivery costs. There are many examples of companies, such as Webvan (Agatz et al. 2011), that became popular and then went bankrupt because they were unable to design a sustainable distribution model.

Obviously, AHD is not limited to e-grocers and appears in several other industries. As mentioned by Campbell and Savelsbergh (2005), this type of distribution is becoming common in drug companies that offer delivery services and companies that deliver office supplies to business customers. It is common in the telecommunications sector, where companies such as AT&T allow customers to book appointments for cable and phone installation through their online channel. It also arises in the context of home health care services (Liu et al. 2014).

In this paper, we focus on the market for commodities such as gas, electricity, and water. In Europe, this market is generally structured in such a way that several trading companies compete against each other to sell the commodities to the customers, while a local distributor is responsible for operating the delivery network. The distributor has to perform several AHD services related to the meters installed at the customer locations. These services involve installing new meters, re-initializing the meter when a customer changes from one trading company to another, closing the meter in case the customer does not pay, performing general maintenance operations, etc.

In particular, we study the activity of a large Italian corporate group called *Iren*, which provides gas, electricity, and water services throughout the country. We address the practical problem faced by *IRETI*, a subsidiary of this group that operates the distribution of gas and water in the region of Emilia Romagna, providing services for approximately 1,300,000 residents. We concentrate on the province of Reggio Emilia, where IRETI performs AHD services by using a dozen of hired technicians that depart from three different depots, as well as a third-party logistics provider.

Following a series of laws from the European Union, starting in 1998 and culminating in the directive 2009/73/CE (European Parliament 2009), the market for natural gas has been regulated and common rules have been established. As a consequence, all distribution companies operating in this market were forced by their national authorities to implement

online applications in which customers are able to make requests for services by booking time slots. These time slots are subject to national regulation constraints and are made available in certain days and time intervals.

Designing an automated approach that is capable of proposing cost-efficient time slot tables is a very complex task, because it involves solving a two-stage stochastic decision problem. The first stage consists in choosing a time slot table for each region, while the second stage consists in routing the technicians after the customers' demand becomes known. To solve the problem we use a three-step algorithm. In the first step, we are interested in producing a time slot table for each region such that the amount of unserved demand is minimized. A key aspect of this step is that the expected demand has to be taken into account so that the offer of slots is neither underestimated nor overestimated. In Step 2, we simulate the fact that customers request services by booking time slots in the time slot table associated with their region, thus revealing the actual demand. Finally, having the assignments of services to time slots, in Step 3 we design the routing plan for each technician such that the total distance traveled is minimized. Notice that a solution for the problem is the actual set of time slot tables generated in Step 1, but it is only possible to evaluate the quality of a given solution by first simulating Step 2, and then evaluating the actual routing costs in the last step. Furthermore, as a consequence of the stochastic nature of the second step, the design of exact algorithms as well as the evaluation of meaningful lower bounds are not straightforward, and heuristic approaches are preferable.Another important aspect of the problem is that service requests cannot be rejected and, eventually, all services have to be performed. This feature is not exclusive to our problem and can be found in other AHD applications, such as in e-groceries, where customers must always receive their purchases.

To address this problem, we propose an optimization method based on the concept of *large neighborhood search* (LNS) that is capable of producing feasible and cost efficient solutions. A key component of our approach is an *integer linear programming* (ILP) model that is able to solve the Step 1 problem by taking into account the expected demand of each region and the popularity of each time slot. The model also considers that an even distribution of consecutive opened time slots is usually beneficial, as it increases the probability of having services assigned to neighbor time slots. These services can be performed consecutively by the same technician, possibly resulting in routing cost savings. The quality of the solutions generated at each LNS iteration is evaluated by a procedure that first reveals customers demands in Step 2 with an algorithm that simulates customer choices in the online application. To this end, we propose four different simulation strategies and the

decision of which one to use is an input of the LNS. Two of them assume that all time slots are equally popular and perform an even distribution of the services among the available slots, whereas the other two are based on past data from the online application. Once all customer requests are assigned to time slots, in Step 3 we must design the routing plan for the technicians. This routing problem involves multiple depots, multiple vehicles and strict time windows, and is solved by means of a second ILP model, which evaluates the optimal routing plan that minimizes the total distance travelled.

The remainder of this paper is organized as follows. A brief review of the related literature is presented in Section 2 and the problem definition is provided in Section 3. The algorithm that we use to simulate customer choices and evaluate the expected solution costs is discussed in Section 4. Our complete solution method is presented in Section 5 and computationally evaluated in Section 6. Finally, in Section 7 some conclusions are drawn and future research directions are proposed.

2. Literature review

There are many challenges and opportunities relevant to operations researchers in the context of AHD. Agatz et al. (2008) use as an illustrative example one of the largest internet grocery stores in the U.S. at the time, named Peapod, to identify combinatorial problems arising in AHD and suggest potential ways to address them.

According to Campbell and Savelsbergh (2005), the fulfillment process of requests for most AHD service models can be divided into three phases: 1. order capture and promise; 2. order sourcing and assembling; and 3. order delivery. There seem to be two main streams of research in the literature on AHD. While some studies concern the management of service time windows, others emphasize the routing and scheduling of delivery tours. Regarding the former, the literature focuses either on the tactical or operational level.

The tactical level is concerned with the design of the time windows themselves, i.e., the number of time slots to offer, their length, and whether or not they overlap. Punakivi and Saranen (2001) assess the impact of variations on the length of time windows in both attended and unattended home deliveries. They show that significant cost reductions can be achieved in a scenario with completely flexible unattended services, compared to attended deliveries with two-hour time windows. Campbell and Savelsbergh (2005) report that an increase in time window length from one hour to two hours can result in a 6% increase in profits. This benefit is even greater when considering an expansion to three-hour time windows, reaching up to 11%. Of course there is an associated trade-off, that is, the longer the time windows the lower the QoS level that is provided.

The operational level refers to the selection of time slots by customers and decisions such as when to open and when to close certain time slots. In this spirit, Campbell and Savelsbergh (2005) propose several mechanisms to determine when to accept or reject requests. In a later work (Campbell and Savelsbergh, 2006), the same authors show that the use of incentive polices to influence customer behavior can result in significant reductions in delivery costs.

In the second stream of research, the basic problem is the *vehicle routing problem with time windows* (VRPTW), which has been extensively studied in the last two decades. For detailed reviews of the literature on this problem, we refer the interested reader to the surveys of Bräysy and Gendreau (2005a,b) and of Baldacci et al. (2012).

Spliet and Gabor (2015) have introduced the *time window assignment vehicle routing problem* (TWAVRP), where time windows have to be assigned to a fixed set of customers before the demand is known. Once the demand is revealed, a routing plan is constructed with the objective of minimizing the travelling costs. They propose a branch-and-price algorithm with two types of route relaxation, and use capacity inequalities to strengthen their lower bound. Computational results indicate that their approach is able to solve instances with up to 25 customers and 3 demand realization scenarios. Our problem is different because the set of customers may vary consistently from one week to another, and because the time windows are not assigned by the company but chosen by the customers.

In the TWAVRP each customer requires a service in each scenario, while in the *consistent vehicle routing problem* (ConVRP), proposed by Groër et al. (2009), this constraint is relaxed. In addition, customers must always be visited by the same driver in each scenario and there is a limit on the total driving time. Recently, Spliet and Desaulniers (2015) proposed the *discrete time window assignment vehicle routing problem* (DTWAVRP). This problem differs from the TWAVRP by considering, for each customer, a finite number of candidate time windows from which a single one must be selected. The authors indicate that this problem is common in retail chains and that they have encountered it while collaborating with Dutch companies in this industry. They developed an exact branch-and-price algorithm and five column generation heuristics that are used to solve large instances when the exact algorithm fails.

In practice, operators might not have the same skill set and cannot perform all the types of services. In such cases, an efficient routing plan depends not only on routing costs, but also on an optimized assignment of operators to services considering their particular skills. Chen et al. (2016) propose an approach for a *technician routing problem* considering the skill set of each operator, and the fact that operators evolve their skills over time as they

learn and gain more experience. The authors show that considering these factors can lead to significant improvements in the quality of the solutions obtained.

To our knowledge, the problem that most resembles the one considered in this paper is the one proposed by Agatz et al. (2011). They study a practical case faced by the Dutch e-grocery company Albert.nl. For each zip code, the company must decide on a set of time slots to offer while minimizing the delivery costs and ensuring an acceptable QoS level to customers. Prior to their study, the design of the time slot tables at Albert.nl was performed manually. Therefore, the authors aimed at developing a fully automated approach to create high quality time slot tables in a short time. Customers can choose the time of their delivery from two-hour time slots having a different fee depending on the day and time of the delivery. This is an interesting feature, common in the e-grocery industry, which can be used to inflect customer behavior. In fact, because of this pricing policy, the authors suppose all time slots to be equally popular and demand to be evenly distributed across the offered slots. Another assumption is that the expected weekly demand for each zip code is known, and it is independent of the time slot table offered. To solve the problem, they propose a continuous approximation and an integer programming algorithm. Routing costs are approximated by a seed-based approach, where customers that are served in the same time slot are clustered and, for each cluster, one of the customers becomes the seed. The actual routing cost is then evaluated as the sum of a route through the seeds plus an estimation of the cost of visiting the customers within each cluster. From the analysis of the computational results, they present some useful insights, for instance: narrow delivery time windows, although convenient for the customers, can lead to an increase of up to 25% in routing costs compared with the alternative of using two-hour time slots; differentiating the offered time slot table by region can lead to cost savings of up to 10%; and optimization of time slot management decisions has a greater impact when the vehicle capacity allows deliveries in several consecutive time slots.

Our contribution cannot be directly compared with that of Agatz et al. (2011) because we consider different policies to simulate customer behaviors and different cost evaluations obtained by designing actual vehicle routes. Although we are not the first to propose an integrated approach for an AHD problem (see, e.g., Cleophas and Ehmke 2014 and Yang et al. 2016), our case study has some peculiarities that prevent us from using many of the methods from the literature. For instance, because of market regulations, we cannot impose different fees for time slots to steer customer choices, and neither are we allowed to apply order acceptance mechanisms, as all requests must be accepted.

Furthermore, to ensure good quality of service, different customer driven policies can

be put in place by the company, such as ensuring that there are not too many consecutive days without any offer of time slots, or that there is a balanced distribution of opened slots between the morning and the afternoon. However, while these policies lead to solutions with a higher quality of service, they also tend to increase the expected costs. In this sense, another contribution of this study is to analyze the trade-off of implementing such policies, and decide whether they can still be profitable for the company.

3. Problem definition

In this section we formally define our optimization problem and present the notation used throughout the paper. The problem comprises three decision steps, which we describe in detail.

Step 1: creation of time slot tables. The aim of this step is to design time slot tables that minimize unserved demand and routing cost while satisfying capacity constraints. We are given a set R of regions, each having an expected total service time. In our case study, services may take either half an hour or one hour, thus we model the expected total service time for a region $r \in R$ by an integer value q_r specifying the number of half hours of service required (i.e., if $q_r = 1$ only half an hour of service is required, if $q_r = 2$ an hour is required, and so on). From now on, let us call q_r the *expected demand* of region r. Let us also use the term basic time resource, or simply *resource* when no confusion arises, to define a time interval of half an hour.

The time horizon is divided into a set D of days, and each day is divided into a set T of non-overlapping time intervals. In our case study, $|R| = 12$ as shown in Figure 1, $|D| = 5$ (corresponding to 5 working days in a week, from Monday to Friday) and $|T| = 9$ (corresponding to intervals [8:30; 9:30], [9:30; 10:30], ..., [16:30, 17:30], where [12:30, 13:30] is reserved for the lunch break). In the following a *time slot* is defined by the pair (d, t).

We are also given a set M of depots. Each region is associated univocally with a depot, so we can set $R = \cup_{i \in M} R_i$, where R_i is the subset of regions associated to depot $i \in M$. Each depot i has Q_i technicians that are available to perform services and must start and end their routes at i. Each technician provides σ resources per time slot. In our case study $|M| = 3$, and $\sigma = 2$ because all time slots last one hour. Because travel times are usually small compared to the actual time spent at the customer locations, we assume that service times include the travel time to the next customer. Note that this assumption is also in use at the company.

A *time slot table* for a region r is an assignment of resources to the time slots. Let u_{rdt} be an integer variable giving the number of resources assigned to region r and time slot

(d, t). A Step 1 solution is a collection of time slot tables, one per region, satisfying the following capacity constraints:

$$\sum_{r \in R_i} u_{rdt} \leq \sigma Q_i \qquad i \in M, d \in D, t \in T, \tag{1}$$

which specify a limit to the number of resources that can be allocated to each time slot per day and depot. Note that there is no constraint forcing the company to allocate resources for all time slots. Therefore, it is possible to create time slot tables that have a subset of slots closed. This feature can be used, for example, to restrict customer choices and to cluster service requests in neighbor slots as an attempt to optimize the final routing costs.

Step 1 has two objectives: the primary one is to minimize the amount of unserved demand, while the secondary one is to minimize the routing cost. The values of these two objectives, for a given Step 1 solution, are computed in Steps 2 and 3, respectively.

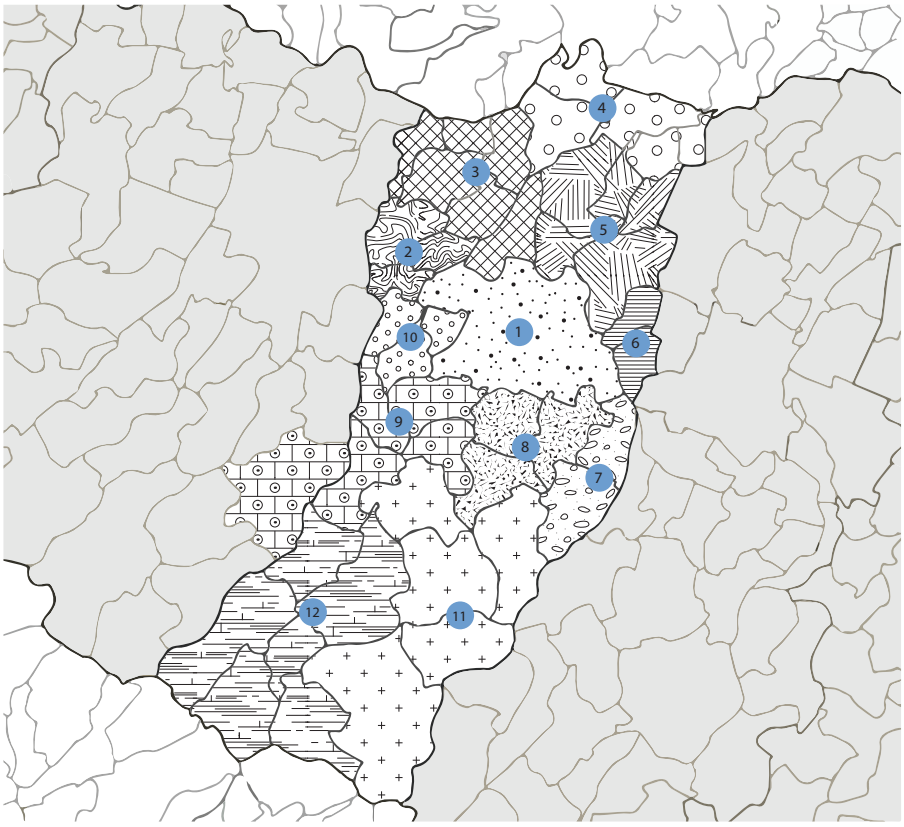


Figure 1: Division of the the territory of action into 12 non-overlapping regions.

Step 2: booking of time slots. In Step 2, the actual demand in a given time horizon is revealed. Let $\bar{u}_{rdt}$ define the Step 1 solution at hand, and let J denote the set of customers now requiring a service. This set is partitioned as $J = \cup_{r \in R} J_r$, where J_r is the subset of customers associated to region r. Each customer j requires a service of w_j resources (with w_j being equal to 1 or 2 in our case study), and attempts to book her *appointment* in the time slot table by choosing a time slot of her liking according to the availability provided by the $\bar{u}_{rdt}$ values.

Let a_{jrdt} be a binary variable taking the value 1 if and only if customer j books her appointment in time slot (d, t) of region r. A Step 2 solution is a set of appointments satisfying the constraints

$$\sum_{j \in J_r} w_j a_{jrdt} \leq \bar{u}_{rdt} \qquad r \in R, d \in D, t \in T, \tag{2}$$

which ensures that the set of appointments for each time slot, day, and region does not consume more resources than those allocated in Step 1.

The decision is made by each customer, on a first-come first-served basis, independently from the will of other customers and of the company. To model customer behavior in the most realistic way, we have thus developed a set of simulation strategies (discussed below in Section 4.1) that we use to set the values taken by the a_{jrdt} variables.

Let us denote the total revealed demand of a region r by $W_r = \sum_{j \in J_r} w_j$. Notice that there is no guarantee that the total revealed demand is equal to the expected demand used in Step 1. In other words, W_r could be either greater than, equal to, or lower than q_r. Consequently, there can be cases in which demand is not fully satisfied. The cost of a Step 2 solution is indeed given by the sum of unserved demands. This value is multiplied by a penalty parameter ρ, which is expressed in kilometers in our case study to make it directly comparable with the routing cost evaluated in Step 3.

Step 3: design of the routing plan. Let $\bar{a}_{jrdt}$ define the set of appointments computed in Step 2. Now let $G = (V, A)$ be a digraph, in which $V = J \cup M$ is the set of vertices and A is the set of arcs. Note that the graph is far from being complete, as A contains only arcs that connect vertices satisfying the chronological order imposed by the appointments (in a way that is made clear in Section 4.2 below). Let c_{hk} be a non-negative routing cost associated to arc $(h, k) \in A$. The aim of Step 3 is to define a set of routes for the technicians satisfying the following constraints:

- each route starts from a depot and returns to the same depot at the end of the services;

- at most Q_i routes are selected for each depot $i \in M$;
- each technician provides at most σ resources for each time slot;
- each appointment is serviced by a technician.

The cost of a Step 3 solution is given by the sum of the routing costs of the selected routes.

Notice that, in Step 3, a technician may perform a service in a region that is not associated with its depot if, by doing so, routing costs are reduced. This it is not considered in Step 1 to avoid creating time slot tables that reflect an unrealistic availability of operators, located too far from their depot to perform services at low cost. Notice also that, in case a route contains one or more time slots without any required activity, then the technician must return to the depot, perform office activities during empty time slots, and then depart again to complete the services of the day.

An interesting detail is that, due to strict regulations imposed by the national authority on the sector, the company cannot influence customer decisions by putting differentiated fees, applying penalties or giving incentives. Consequently, there may be time slots significantly more popular than others, and many policies to reduce operational costs, common in other AHD problems (see, e.g., Campbell and Savelsbergh 2006), cannot be used.

Furthermore, we note that the company is able to outsource a certain number of services through a third-party operator at a fixed cost per service, thus avoiding paying some of the penalties. With the aim of not further complicating the problem, we disregard decisions about which and how many services to outsource. Instead, we approximate outsourcing costs in a simple way by using the aforementioned penalty ρ for any unserved demand.

4. Evaluating the solution cost

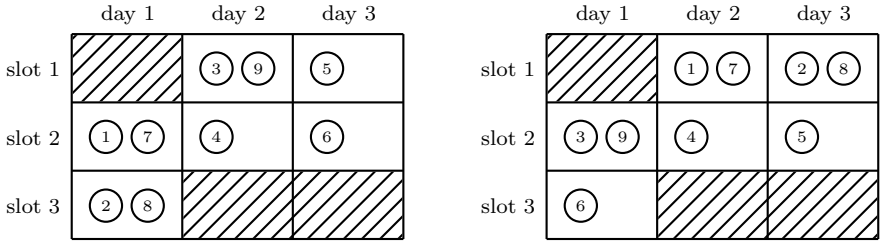
To evaluate the expected cost of a given solution, the customers have to first book their time slots. Only then, routing and penalty costs can be estimated. This is a difficult task, because the behavior of customers is not known beforehand and cannot be predicted with total accuracy. However, it is possible to simulate different scenarios according to a range of factors, including the use of historical data regarding the usage of the online application. In this section, we first propose four simulation strategies and then present an ILP model to evaluate the routing costs.

4.1. Simulation strategies

We are given a time slot table, decided in Step 1 of our approach, and a set of customers that require a service. The time horizon is a week (5 working days and 8 available time slots per day). The aim of the simulation process is to fix appointments in the time slot table by mimicking the customer behavior under reasonable assumptions.

The first two strategies that we have adopted are called *evenly vertical* (EV) and *evenly horizontal* (EH). They are based on the assumption that all time slots are equally popular and services can be distributed evenly throughout the available time slots, as done in Agatz et al. (2011). At each iteration we choose a service and assign it to the next slot that has enough capacity to accommodate it. Except for the first iteration, which starts from the beginning of the time slot table, the others begin the search for a slot right after the one where the previous assignment was made. Notice that, during the assignment of a service, it might happen that the end of the table is reached while searching for an available slot. In this case, we return to the beginning of the time slot table and continue the search. In case there is no feasible slot to insert a service, we consider it as being unserved.

The difference between EV and EH lies in the criteria used for choosing the next slot. While EV performs a vertical search, looking at the remaining time slots on the same day before proceeding to the next day, in EH a horizontal search is used instead. As an illustrative example, consider a scenario where there are 9 services to be assigned to a small time slot table composed by just 3 days and 3 time slots, as the one in Figure 2-(a). Each circle represents a single service and the number inside it specifies the order of assignment. For simplicity, consider that only non-hatched time slots are opened, that a single technician is available to perform $\sigma = 2$ services in each opened slot, and that each service requires 1 unit of resource. Note that closed time slots were chosen arbitrarily for the purpose of this example. Figure 2-(a) shows the result obtained by using EV, and Figure 2-(b) the result obtained by using EH.



(a) evenly vertical (b) evenly horizontal

Figure 2: Examples of simulation strategies *evenly vertical* (EV) and *evenly horizontal* (EH)

Our instances are based on historical data from the company, and information on the actual time slots for which services were performed is available. However, this information cannot be used by EV and EH. Our third strategy, called *rescheduling based* (RB), overcomes this issue by using information on the real assignments of services when simulating customer choices as follows. Let $(\tilde{d}, \tilde{t})$ be the time slot that was actually chosen in real life by a customer $j \in J$. At each iteration we select a customer j and try to assign her service to a time slot close to $(\tilde{d}, \tilde{t})$. In even iterations, we search for a feasible time slot $(\tilde{d}, t)$, i.e., we keep the day fixed but accept a different time. In odd iterations, we search for a time slot $(d, \tilde{t})$, i.e., at the same time but possibly in a different day. In case the process fails, we use the EV strategy to find a feasible time slot. If no time slot with enough capacity is available, then we mark the customer as unserved.

For the last strategy, called *popularity based* (PB), we first evaluated the popularity of each time slot on the period from 2010 to 2013. To this aim, we computed the frequency at which a time slot (d, t) was chosen, and then normalized the values, thus obtaining an array of probabilities p_{dt} such that $\sum_{d \in D} \sum_{t \in T} p_{dt} = 1$. At each iteration a customer $j \in J$ is chosen and then assigned to a time slot (d, t) having enough residual capacity, if any, according to probabilities p_{dt}.

For each unserved service a penalty ρ is paid. The resulting penalty cost is added to the routing cost, which is calculated when designing the routing plan.

4.2. Design of a routing plan

Having simulated the customer choices, we must elaborate a routing plan for each technician with the aim of minimizing total travelling costs. The associated routing problem is decomposable by day, since the routing plan of one day is independent from those of the other days, and can be modeled as a variant of the *multidepot multiple traveling salesman problem* (MmTSP).

The MmTSP is a variant of the well known traveling salesman problem that consists in finding tours for salesmen starting from different depots in such a way that all customers are visited exactly once and the total traveled distance is minimized. If salesmen are constrained to end their routes at the same depot they originated from, the problem is called *fixed destination* MmTSP (f-MmTSP), otherwise it is referred to as *non-fixed destination* MmTSP (nf-MmTSP). In the MmTSP literature, an upper limit on the total distance traveled by each technician is also usually imposed, but this is not required in our problem. We refer the interested reader to Bektas (2006) and to Kara and Bektas (2006).

At Step 3, customers have already chosen their appointments in the available time slots,

so they must be visited in a way that respects the time window constraints. Because these time windows are discretized in intervals that do not overlap, it suffices to solve the MmTSP under a *time-extended network* containing only those arcs that do not violate the time windows. To this aim, let us add to the original set of time slots two dummy slots, 0 and $|T|+1$ representing, respectively, the beginning and the end of a working day. The resulting set of time slots is thus $T' = \{0, 1, ..., |T|, |T|+1\}$. Let J'_t be the set of customers that chose an appointment at time $t \in T$. Let us also duplicate each depot $i \in M$ in a set $M_i = \{i_0, i_1, ..., i_{|T|+1}\}$ of copy vertices, where each vertex i_t represents a visit to i at time t.

Now, let us define our time-extended network as a digraph $G' = (V', A')$, where $M' = \cup_{i \in M} M_i$ and $V' = J \cup M'$. The set A' contains two types of arcs. The first type comprises arcs (h, k) connecting each vertex $h \in J'_t \cup \{i_t : i \in M\}$ to each vertex $k \in J'_{t+1} \cup \{i_{t+1} : i \in M\}$, for $t \in T' \setminus \{|T|+1\}$, ensuring that h and k are not two different depots. The second type comprises arcs (h, k) connecting two customers $h, k \in J'_t$ (for $t \in T$) for which $w_h + w_k \leq \sigma$ holds.

An illustrative example is shown in Figure 3, where $M = \{1, 2\}$, $T = \{1, 2, 3\}$, $J'_1 = \{3, 4\}$, $J'_2 = \{5, 6\}$, $J'_3 = \{7, 8\}$ and $\sigma = 2$. All customers require two units of resource, except for customers 7 and 8 that require one unit each. Therefore, customers 7 and 8 are the only ones that can be served by the same technician in the same time slot. Note that the graph is not entirely acyclic because multiple customers can be served in the same time slot, and consequently the problem cannot be directly modeled as a multi-depot vehicle scheduling problem (see, e.g., Desaulniers et al. 1998).

Recall that c_{hk} is the routing cost of arc (h, k) and that each depot $i \in M$ has Q_i technicians. Let binary variable x_{hk} take the value 1 if arc $(h, k) \in A'$ is used and 0

otherwise. The nf-MmTSP can be modeled as

$$\text{(nf-MmTSP)} \quad \min \; z_{nf} = \sum_{(h,k)\in A'} c_{hk} x_{hk} \tag{3}$$

subject to

$$\sum_{k\in V'} x_{i_0 k} \le Q_i \qquad \forall\, i \in M \tag{4}$$

$$\sum_{h\in V'} x_{hj} = 1 \qquad \forall\, j \in J \tag{5}$$

$$\sum_{k\in V'} x_{hk} - \sum_{k\in V'} x_{kh} = 0 \qquad \forall\, h \in V' \setminus \{i_0, i_{|T|+1} : i \in M\} \tag{6}$$

$$\sum_{h\in \bar{S}} \sum_{k\in S} x_{hk} \ge \left\lceil \frac{\sum_{h\in S} w_h}{\sigma} \right\rceil \qquad \forall\, S \subseteq J'_t, \bar{S} \in J'_{t-1} : t \in T \tag{7}$$

$$x_{hk} \in \{0,1\} \qquad \forall\, (h,k) \in A'. \tag{8}$$

The objective function (3) minimizes the total routing cost. Constraints (4) impose the maximum number of technicians for each depot, whereas constraints (5) specify that all customers must be visited exactly once. Flow conservation is imposed by constraints (6), for all nodes except for the copies of the depots at times 0 and $|T|+1$. Subtour elimination and capacity constraints are needed only for subsets of customers in the same time slot, and are imposed by means of constraints (7). We note that constraints (7) are separated only at integer points by the enumeration and inspection of all possible paths.

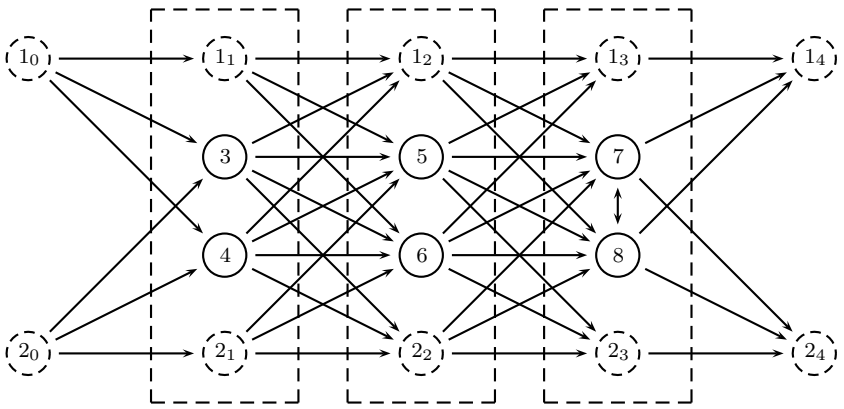


Figure 3: Example of a time-extended network with 2 depots (1 and 2), 6 customers and 3 time slots

To adapt formulation (3)-(8) to the fixed problem variant, f-MmTSP, it suffices to add the following set of infeasible path constraints:

$$\sum_{(h,k)\in\xi} x_{hk} \leq |\xi| - 1 \qquad \forall\ \xi \in \mathcal{R}, \tag{9}$$

where $\mathcal{R}$ is the set of all routes that begin and end at different depots, and, for every route $\xi \in \mathcal{R}$, $|\xi|$ denotes the number of arcs in the route.

Considering that this routing problem has to be solved every time we need to evaluate the quality of a solution for the main problem, efficiency is of great importance. We performed computational experiments with both variants and noticed a significant loss in performance with the inclusion of constraints (9) in the formulation, while the gap between the two solution values was usually very small. Therefore, we decided to use the nf-MmTSP to guide the search of our LNS.

5. Solution methodology

The practical nature of the problem suggests the use of efficient heuristic approaches, able to provide good quality solutions in a reasonable amount of time. In addition, the use of exact algorithms would not be straightforward because there is no deterministic method to accurately evaluate the cost of a solution. This is due to the fact that customer choices play a crucial role in determining the final routing costs and is an aspect that cannot be determined beforehand.

An interesting feature of our problem is that the search space is very large. One must decide how many resources to offer for each time slot, day and region, while satisfying operational constraints. We note that, although it is usually fairly easy to find feasible solutions, it is computationally expensive to evaluate their expected cost because, as described in Section 4, this requires the solution of an ILP.

For these reasons, we have decided to adopt the LNS framework, originally proposed by Shaw (1998) and later used successfully in many routing problems (see, e.g., Pisinger and Ropke 2010). The LNS framework relies on *destroy* and *repair* methods. While the former destroy parts of the solution, the latter are responsible for regaining feasibility. Algorithm 1 shows the pseudocode of our LNS. The algorithm receives in input the expected demand q_r of each region $r \in R$ ($regionsDemand$), an initial solution ($initialSolution$), the simulation strategy to be used ($simulationStrategy$), and the number of iterations to be performed ($numIt$). In our implementation, the initial solution is generated by an ILP that will be

explained in detail later in Section 5.2. In lines 2 and 3 the local variables are initialized and the cost of the initial solution is evaluated. At each iteration of the main loop (line 4), a candidate solution (*newSolution*) is generated by destructing part of *bestSolution*, and then invoking the repair method (line 5). If the newly created solution has a better expected cost than the incumbent, then *bestSolution* and *bestCost* are updated accordingly (line 7). The algorithm stops after *numIt* iterations and the incumbent solution is returned. In our implementation we set $numIt = 50$.

Algorithm 1 Pseudocode of our LNS algorithm

```
1:  function LNS(regionsDemand, initialSolution, simulationStrategy, numIt)
2:      bestSolution ← initialSolution
3:      bestCost ← estimateSolutionCost(bestSolution, simulationStrategy)
4:      for it ← 0 to numIt do
5:          newSolution ← repair(destroy(bestSolution), regionsDemand)
6:          cost ← estimateSolutionCost(newSolution, simulationStrategy)
7:          if cost < bestCost then updateBestSolution(newSolution, cost)
8:      end for
9:      return bestSolution
10: end function
```

In the following, we describe the proposed destroy and repair methods and explain how we generate a feasible initial solution.

5.1. Destroy method

As mentioned by Pisinger and Ropke (2010), the choice of the destroy method is an important aspect for the effectiveness of an LNS algorithm. If the degree of destruction is rather weak and only a small portion of the solution is destroyed, then the search space explored is considerably limited and the benefits of using such a heuristic may be lost. Similarly, if the degree of destruction is too high, then the heuristic tends to degrade into a repeated re-optimization process.

In our destroy method, for each region $r \in R$ we randomly select β opened time slots (i.e., β slots that have at least one unit of resource assigned) and empty them. Notice that if there are β or fewer opened time slots, then the table is totally destroyed. In our experiments, where each time slot table has a total of 40 time slots (5 days and 8 usable time slots per day), we found good results by using $\beta = 10$.

5.2. Repair method

Once a solution is partially or totally destroyed it must be reconstructed. To this end, we developed a repair method based on an ILP model. The basic idea is to use a *fitness function* that tries to maximize the number of served demands and the popularity of the chosen slots, but at the same time penalizes features that may have a negative impact on the routing plan.

Recall that $J_r \subseteq J$ is the set of customers associated with region r, that $p_{dt} \in [0,1]$ defines the probability that a customer chooses a time slot (d,t) (see Section 4.1), and that variable u_{rdt} gives the number of resources allocated to time slot (d,t) in region r (see Section 3). Let also $\ell \in T$ be the time slot dedicated to the lunch break.

During experimental analysis, we noticed that an even distribution of resources among consecutive time slots is a key factor in minimizing the routing costs. This allows technicians to perform longer routes and make fewer trips to the depots. While this is not strictly necessary to ensure feasibility, it can greatly improve the quality of the solutions generated. We incorporate this observation into the model by defining a penalty for uneven distribution of resources among consecutive time slots. Let v_{rdt} be the number of displaced resources for a given time slot (d,t) of region r. This value is evaluated considering either a forward or a backward approach. In the forward approach, the number of displaced resources in (d,t) is $v_{rdt} = |u_{rdt} - u_{rd,t+1}|$, in other words, v_{rdt} gives the difference in the number of allocated resources between (d,t) and $(d,t+1)$. Similarly, in the backward approach $v_{rdt} = |u_{rdt} - u_{rd,t-1}|$ is the difference between the amount of resources allocated in (d,t) and $(d,t-1)$. Let γ be the penalty associated with each unit of displaced resource.

To model cases where the required demand cannot be entirely satisfied by the available technicians, let z_r be an integer variable specifying the amount of demand that could not be allocated to region $r \in R$ due to capacity constraints. In addition, let Ω be a penalty value for each unassigned demand unit. We are now ready to introduce the following *three-index formulation* (TIF):

$$\text{(TIF)} \quad \max \; f_{TIF} = \sum_{r\in R}\sum_{d\in D}\sum_{t\in T} p_{dt}u_{rdt} - \sum_{r\in R}\sum_{d\in D}\sum_{t\in T}\gamma\, v_{rdt} - \sum_{r\in R}\Omega z_r \tag{10}$$

subject to

$$\sum_{d\in D}\sum_{t\in T} u_{rdt} = q_r - z_r \qquad \forall\, r \in R \tag{11}$$

$$\sum_{r\in R_i} u_{rdt} \leq \sigma Q_i \qquad \forall\, i \in M, d \in D, t \in T \tag{12}$$

$$v_{rdt} \geq u_{rdt} - u_{rd,t+1} \qquad \forall\, r \in R, d \in D, t \in T \setminus \{\ell - 1, \ell, |T|\} \tag{13}$$

$$v_{rdt} \geq u_{rd,t+1} - u_{rdt} \qquad \forall\, r \in R, d \in D, t \in T \setminus \{\ell - 1, \ell, |T|\} \tag{14}$$

$$v_{rdt} \geq u_{rdt} - u_{rd,t-1} \qquad \forall\, r \in R, d \in D, t \in \{\ell - 1, |T|\} \tag{15}$$

$$v_{rdt} \geq u_{rd,t-1} - u_{rdt} \qquad \forall\, r \in R, d \in D, t \in \{\ell - 1, |T|\} \tag{16}$$

$$u_{rd\ell} = 0 \qquad \forall\, r \in R, d \in D \tag{17}$$

$$u_{rdt} \geq 0, \text{integer} \qquad \forall\, r \in R, d \in D, t \in T \tag{18}$$

$$v_{rdt} \geq 0, \text{integer} \qquad \forall\, r \in R, d \in D, t \in T \tag{19}$$

$$z_r \geq 0, \text{integer} \qquad \forall\, r \in R. \tag{20}$$

The fitness function (10) maximizes the serviced demand and the aggregated value of popularity per allocated resource and, at the same time, tries to avoid uneven distribution of resources. Constraints (11) specify that the number of allocated resources per region is equal to the served demand. Recall that in Step 1 each region is associated with a single depot and each technician can provide at most σ resources per time slot. Constraints (12) ensure that the number of resources allocated to each time slot does not exceed the total technician capacity. Constraints (13) and (14) evaluate the penalties for uneven distribution of resources using a forward approach. Notice that this approach is not feasible for the time slot just before the lunch break and the last one of the day. For these slots, a backward approach is used instead in (15) and (16). Constraints (17) impose that time slots associated with the lunch break are always closed. Finally, constraints (18)-(20) formally define variables u, v, and z.

Through computational experiments, formulation TIF has proved to be a good tool to generate feasible solutions for the problem. Thus it is first used as a method for generating initial solutions for the LNS algorithm. Its main use is, however, as a repair method. Because the main objective of the repair method is that of producing feasible solutions, but not optimizing the whole scenario, we impose in TIF additional constraints specifying that the number of resources allocated to time slots not altered during the call to the destroy method can only increase. Let Θ_r be the set of time slots not affected by the destroy procedure in region $r \in R$ and b_{rdt} the amount of resources in time slot $(d, t) \in \Theta_r$. Then, at every call to the repair method we add to TIF the following constraints:

$$u_{rdt} \geq b_{rdt} \qquad \forall\, r \in R, d \in D, t \in T : (d, t) \in \Theta_r. \tag{21}$$

5.3. Quality of service

In AHD it is important to establish a certain *quality of service* (QoS) level in the time slot tables offered to customers. As mentioned by Agatz et al. (2011), an important aspect

is the balanced distribution of resources between morning and afternoon time slots along the week. Morning slots are those with an index in T smaller than ℓ, and afternoon slots are those with an index greater than ℓ. Let α_m and α_a be, respectively, the minimum percentage of resources that should be allocated to morning and afternoon slots, satisfying $\alpha_m + \alpha_a \leq 1$. We first impose a high QoS by embedding into TIF the following constraints:

$$\sum_{t=1}^{\ell-1} \sum_{d \in D} u_{rdt} \geq \alpha_m q_r \qquad \forall\, r \in R \tag{22}$$

$$\sum_{t=\ell+1}^{|T|} \sum_{d \in D} u_{rdt} \geq \alpha_a q_r \qquad \forall\, r \in R. \tag{23}$$

The second type of QoS constraints that we consider impose a limit g on the maximum number of consecutive days without any time slot opened in a region. This is particularly useful for time slot tables in regions with low demand. Let y_{rd} be a binary variable indicating whether or not there is at least one time slot opened in day $d \in D$ for region $r \in R$. In addition, let $\phi(d, g) = \{(i \bmod |D|) + 1 : i = d - 1, d, \ldots, d + g - 1\}$. Function ϕ is used to switch from one week to the next one. For instance, if $D = \{1, ..., 5\}$, $d = 4$ and $g = 2$, then $\phi(4, 2) = \{4, 5, 1\}$. Hence, a high QoS is imposed by adding to TIF the following constraints:

$$\sum_{t \in T} u_{rdt} \geq y_{rd} \qquad \forall\, r \in R, d \in D \tag{24}$$

$$\sum_{l \in \phi(d,g)} y_{rl} \geq 1 \qquad \forall\, r \in R, d \in D \tag{25}$$

$$y_{rd} \in \{0, 1\} \qquad \forall\, r \in R, d \in D. \tag{26}$$

From now on we refer to the formulation composed by (10)-(20) and (22)-(26) as TIF$^+$.

6. Computational experiments

Our algorithms were implemented in C++ and the computational experiments were run on a PC with an Intel Core i7-3770 3.40 GHz processor and 8 Gb of RAM. Cplex 12.6.2 was used to solve the ILP models with 8 threads and the default options. We used a set of 52 benchmark instances referring to the weeks of 2012, the year in which the company first introduced the online application to customers. These instances include real distance values, computed with a geographical information system using a map from OpenStreetMap.

6.1. Demand estimation

Currently the company manually generates a solution (i.e., a set of time slot tables) based on a certain estimation of the expected demand (called *base demand* below). This solution is deployed in the online application and is used for the whole year. Then, whenever the demand of services proves to be higher than anticipated, manual adjustments are performed to offer additional slots.

In order to better understand the evolution of the demand along the year, we analyzed the services performed in the period between 2010 and 2013. The results are depicted in Figure 4, where the x-axis represents the weeks of 2012 (year of the introduction of the online application), and the y-axis the aggregated weekly demand expressed in terms of total number of required resources (i.e., $\sum_{r \in R} q_r$). Let *base* be the base estimation of demand by the company in number of resources. Consider *2012* as the weekly demand of resources during 2012, and let *avg-2010-2013* be the average monthly demand during the period from 2010 to 2013. Notice that, in the majority of cases, the actual demand considered by the company to generate their solution is much lower with respect to 2012 and the average from 2010 to 2013. As a consequence, their solution probably needs constant manual adjustments to keep up with the demand.

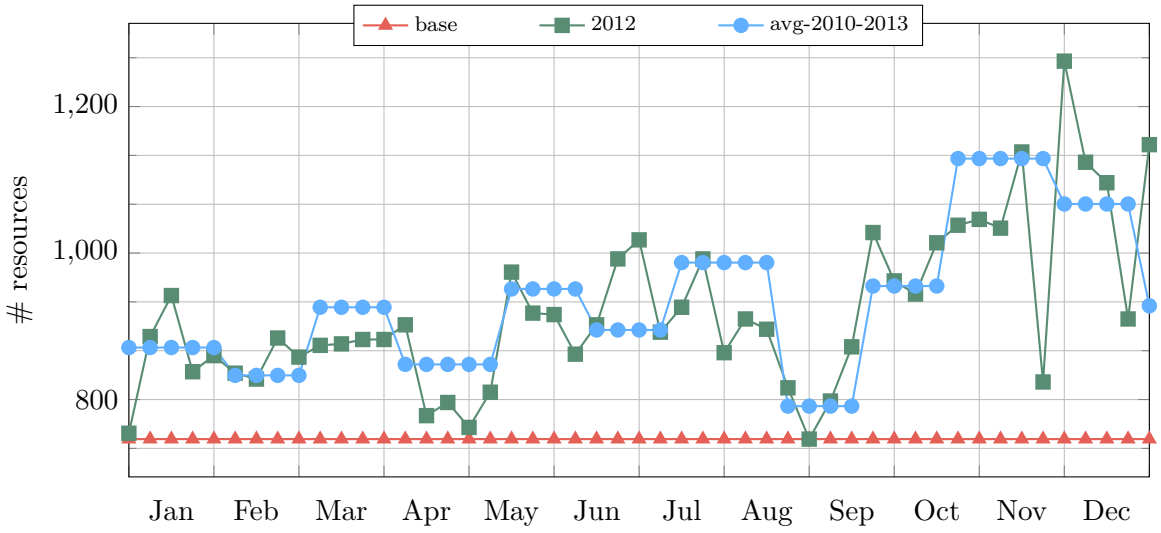


Figure 4: Evolution of the demand in 2012, compared with base demand used by the company and average demand in 2010-2013.

The expected demand of each region is a key input of the LNS, because it is used by formulation TIF to reconstruct solutions, and directly influences the number of resources

offered in the time slot tables. In our experiments we tested the LNS under three demand scenarios. In the first one, called *base demand*, we set q_r (i.e., the expected demand of region r) for $r \in R$, to be the same number of resources as offered in the time slot tables currently employed at the company. In other words, this scenario allows us to evaluate whether the LNS is able to find solutions under the same demand conditions as in the company's solution. The second scenario, referred as *avg demand*, considers instead q_r to be the average demand per month, evaluated considering the period from 2010 to 2013. Lastly, we consider a *utopic* scenario, namely *a priori demand*, in which the number of resources requested for each week is known beforehand and time slot tables are designed accordingly. Although this scenario should provide the best results in terms of minimizing both unserved demand and routing costs, *avg demand* is realistic and can be more easily applied in practice.

Recall that we assume a penalty ρ to be paid for each service that is not assigned during the simulation of customer choices. We set $\rho = 20$ on the basis of considerations from the real world application. In addition, we set the penalty values $\gamma = 0.3$ and $\Omega = 100$ to minimize, respectively, uneven distribution of slots and unassigned demand in formulation TIF.

6.2. Computational performance of LNS

We evaluate our LNS algorithm under the three demand scenarios and compare the results with the expected cost of the solution currently being used by the company. This evaluation is performed by taking the company's time slot tables and evaluating the expected costs by running the algorithm presented in Section 4, under one of the four proposed simulation strategies. Note that, even though we have been provided with the company's solution (i.e., their time slots tables), we cannot directly compute the routing costs incurred by their solution because we do not have the complete information that is needed. For instance, we do not know which paths have been taken and neither which operator performed each service. In any case, because the technicians are experienced drivers and have a good knowledge of the area where they operate, we assume that their routes are fairly close to the shortest paths that we computed.

In Tables 1, 2, 3, and 4 we present the results of a monthly analysis during 2012 for the strategies EV, EH, RB, and PB, respectively. The number of instances per month is shown in column # and the expected cost of the company's solution is shown in column *current*. Notice that, because the evaluation of cost depends on the chosen simulation strategy, these values might be different from one table to the other. Columns named

cost specify the average expected cost in kilometers of the time slot tables for the set of instances of the respective month. Let $cost_{current}$ be the value in column *current*, and $cost_{LNS}$ be the expected cost of the respective LNS approach. The average improvements, shown in column *gap*, are evaluated as $((cost_{current} - cost_{LNS})/cost_{current}) \times 100$. Finally, the average percentage of unassigned demand is presented in columns named d_{un} and the average computing times in seconds are shown in columns named *sec*.

Table 1: Evaluation of LNS for the simulation strategy EV

		current	LNS - base demand				LNS - avg. demand				LNS - a priori demand			
month	#	cost	cost	gap	d_{un}	sec	cost	gap	d_{un}	sec	cost	gap	d_{un}	sec
Jan	5	5878.9	5563.3	-5.4	17.7	278.3	5469.0	-7.0	7.3	127.9	5225.4	-11.1	2.6	259.4
Feb	4	5446.4	5274.5	-3.2	16.2	273.1	5002.4	-8.2	5.4	280.0	4882.5	-10.4	2.1	251.7
Mar	4	6110.0	6004.2	-1.7	22.0	282.4	5463.0	-10.6	2.3	274.0	5416.0	-11.4	3.0	284.4
Apr	5	5090.8	4863.3	-4.5	10.5	275.5	4863.3	-4.5	4.3	281.4	4703.0	-7.6	2.0	262.9
May	4	6648.9	6326.6	-4.8	27.4	278.8	5706.2	-14.2	5.7	291.3	5547.7	-16.6	2.4	299.3
Jun	4	6908.0	6724.4	-2.7	32.2	281.6	6285.5	-9.0	13.1	284.7	5803.2	-16.0	2.5	309.2
Jul	5	6570.5	6365.0	-3.1	27.1	286.9	5788.1	-11.9	3.6	290.3	5671.9	-13.7	2.1	290.7
Aug	4	4887.8	4545.9	-7.0	10.1	271.8	4623.2	-5.4	6.3	269.1	4393.6	-10.1	1.6	285.6
Sep	4	7212.1	6989.2	-3.1	36.5	281.5	6355.5	-11.9	9.9	207.4	5923.6	-17.9	3.3	259.5
Oct	5	7672.0	7458.9	-2.8	41.3	281.6	6487.0	-15.4	8.3	49.6	6398.4	-16.6	6.6	217.3
Nov	4	8835.2	8710.6	-1.4	57.2	280.9	7285.4	-17.5	13.7	220.8	7036.3	-20.4	10.0	163.3
Dec	4	6659.1	6617.4	-0.6	37.7	269.7	5919.5	-11.1	14.0	280.4	5427.2	-18.5	5.1	192.1
avg		6493.3	6287.0	-3.4	28.0	278.5	5770.7	-10.6	7.8	238.1	5535.7	-14.2	3.6	256.3

Table 2: Evaluation of LNS for the simulation strategy EH

		current	LNS - base demand				LNS - avg. demand				LNS - a priori demand			
month	#	cost	cost	gap	d_{un}	sec	cost	gap	d_{un}	sec	cost	gap	d_{un}	sec
Jan	5	6002.0	5592.3	-6.8	17.4	283.1	5472.7	-8.8	7.3	174.2	5253.4	-12.5	2.3	269.3
Feb	4	5687.7	5323.2	-6.4	16.4	278.2	5131.1	-9.8	5.7	280.3	4951.1	-12.9	2.0	268.0
Mar	4	6339.6	6056.2	-4.5	21.6	280.9	5570.8	-12.1	2.5	277.1	5472.7	-13.7	2.3	286.0
Apr	5	5297.6	4968.1	-6.2	10.5	280.5	4970.1	-6.2	3.9	281.8	4804.4	-9.3	1.6	282.7
May	4	6828.2	6435.4	-5.8	27.8	284.8	5880.1	-13.9	5.9	289.5	5634.0	-17.5	2.1	277.2
Jun	4	7257.1	6801.1	-6.3	32.6	283.4	6433.5	-11.3	12.9	285.2	5899.4	-18.7	2.2	294.6
Jul	5	6825.0	6441.4	-5.6	26.9	286.0	5991.0	-12.2	3.8	289.9	5805.1	-14.9	1.9	272.9
Aug	4	5120.6	4625.3	-9.7	9.8	277.1	4803.0	-6.2	6.3	282.0	4519.9	-11.7	1.1	275.6
Sep	4	7402.3	7075.5	-4.4	36.2	282.4	6479.2	-12.5	10.0	241.6	6043.2	-18.4	3.0	219.3
Oct	5	7901.4	7516.5	-4.9	41.5	282.5	6601.9	-16.4	8.3	51.7	6529.1	-17.4	6.4	186.6
Nov	4	8976.3	8719.4	-2.9	56.8	282.4	7450.0	-17.0	13.9	198.2	7078.2	-21.1	9.1	139.8
Dec	4	6939.1	6693.9	-3.5	37.4	276.0	6065.8	-12.6	13.7	280.6	5584.6	-19.5	5.1	167.6
avg		6714.7	6354.0	-5.6	27.9	281.4	5904.1	-11.6	7.8	244.3	5631.2	-15.6	3.3	245.0

Results show that our LNS is able to achieve significant improvements in a reasonable amount of time. These improvements range in the intervals 3.4%-14.2% for EV, 5.6%-15.6%

Table 3: Evaluation of LNS for the simulation strategy RB

		current	LNS - base demand				LNS - avg. demand				LNS - a priori demand			
month	#	cost	cost	gap	d_{un}	sec	cost	gap	d_{un}	sec	cost	gap	d_{un}	sec
Jan	5	6167.3	5773.2	-6.4	17.6	275.7	5777.6	-6.3	6.9	115.8	5375.3	-12.8	0.5	269.1
Feb	4	5834.0	5576.8	-4.4	16.6	272.6	5312.2	-8.9	4.8	282.7	5109.6	-12.4	0.6	252.0
Mar	4	6635.5	6298.2	-5.1	21.8	283.4	5891.1	-11.2	2.1	252.9	5619.9	-15.3	0.2	280.7
Apr	5	5650.9	5203.3	-7.9	10.1	264.9	5370.4	-5.0	3.6	278.8	5071.9	-10.2	0.5	270.4
May	4	7168.7	6665.0	-7.0	27.7	273.4	6106.3	-14.8	5.2	293.5	5762.1	-19.6	0.6	268.2
Jun	4	7413.4	7087.7	-4.4	32.6	276.8	6764.1	-8.8	13.3	276.0	6245.9	-15.7	0.9	295.3
Jul	5	6976.1	6684.8	-4.2	27.2	281.2	6355.2	-8.9	3.7	278.2	6017.6	-13.7	0.4	242.2
Aug	4	5362.3	4995.6	-6.8	9.9	270.1	5113.6	-4.6	6.2	275.9	4846.9	-9.6	0.2	274.4
Sep	4	7784.7	7363.1	-5.4	36.5	263.6	6791.7	-12.8	9.9	238.2	6165.0	-20.8	1.5	234.0
Oct	5	8071.2	7788.8	-3.5	41.9	274.8	7001.0	-13.3	7.9	54.2	6653.3	-17.6	5.0	196.0
Nov	4	9246.2	8909.8	-3.6	57.3	279.8	7797.0	-15.7	13.9	184.2	7293.6	-21.1	7.8	152.0
Dec	4	7212.3	6935.9	-3.8	38.3	258.5	6369.9	-11.7	14.3	280.8	5826.0	-19.2	4.2	174.2
avg		6960.2	6606.8	-5.2	28.1	272.9	6220.8	-10.2	7.6	234.3	5832.3	-15.7	1.9	242.4

Table 4: Evaluation of LNS for the simulation strategy PB

		current	LNS - base demand				LNS - avg. demand				LNS - a priori demand			
month	#	cost	cost	gap	d_{un}	sec	cost	gap	d_{un}	sec	cost	gap	d_{un}	sec
Jan	5	6100.3	5683.1	-6.8	16.0	269.1	5695.0	-6.6	6.1	151.2	5464.0	-10.4	1.2	273.4
Feb	4	5907.2	5551.1	-6.0	16.0	276.0	5341.3	-9.6	4.4	281.5	5187.7	-12.2	1.0	255.9
Mar	4	6544.6	6212.7	-5.1	19.8	258.4	5924.4	-9.5	2.1	268.1	5644.1	-13.8	0.7	282.3
Apr	5	5665.5	5152.1	-9.1	9.0	268.8	5334.3	-5.8	3.7	283.1	5130.6	-9.4	1.1	258.1
May	4	7023.4	6577.5	-6.3	26.3	278.9	6073.9	-13.5	4.7	292.6	5883.1	-16.2	1.3	270.2
Jun	4	7409.2	6882.4	-7.1	31.0	276.0	6719.9	-9.3	12.2	263.6	6310.9	-14.8	1.2	246.3
Jul	5	7016.8	6541.2	-6.8	25.5	280.5	6400.1	-8.8	3.4	267.2	6157.2	-12.3	1.0	264.1
Aug	4	5336.3	4812.7	-9.8	8.7	272.4	4987.3	-6.5	5.5	281.8	4866.9	-8.8	1.0	279.2
Sep	4	7607.8	7227.2	-5.0	34.5	274.4	6683.1	-12.2	8.9	239.0	6312.7	-17.0	1.5	225.5
Oct	5	8082.3	7672.5	-5.1	39.9	280.5	7013.5	-13.2	7.5	52.5	6708.9	-17.0	5.2	193.5
Nov	4	9074.1	8759.6	-3.5	54.9	279.2	7753.6	-14.6	13.1	244.6	7295.7	-19.6	8.0	144.5
Dec	4	7125.9	6763.4	-5.1	36.1	272.7	6256.0	-12.2	13.2	284.6	5750.6	-19.3	4.2	168.9
avg		6907.8	6486.3	-6.3	26.5	273.9	6181.9	-10.2	7.1	242.5	5892.7	-14.2	2.3	238.5

for EH, 5.2%-15.7% for RB, and 6.3%-14.2% for PB. They are quite consistent, showing that the algorithm is robust and it is able to adapt to different scenarios. Notice that during high demand periods, such as from September to December, improvements are the highest. This is explained by the high number of requests per time slot in certain regions, which allows for greater routing options and a better optimization of the routing plan. An important aspect in this regard is the fact that formulation TIF usually leads to an even distribution of resources in sequential time slots.

We also note that the average amount of unassigned demand for all strategies on the

most restrictive demand base scenario is very high. Consequently, these solutions rely a lot on the third-party logistics operator. Instead, the *avg demand* scenario seems to provide a good compromise in this regard, as the average of unassigned demand is much lower, ranging from 7.1% to 7.8%, thus requiring much less manual updates. Note that achieving 0% of unassigned demand might not be possible even on the *a priori* scenario, and especially during high demand months such as November. This is mainly due to the availability of operators, which defines the daily capacity of the system.

Table 5 presents the average results of all four simulation strategies to provide a clear overview. Overall, the results indicate that improvements of about 5% on average can be achieved in the most restrictive demand scenario, while in the *a priori* case, where the demand is known beforehand, these improvements reach up to 15%, on average.

Furthermore, we performed additional experiments and noticed that, giving the LNS the time slot tables used by the company as initial solution resulted in a small but consistent worsening of the solution costs from 2.7% up to 5%. This shows that using TIF as the method for generating initial solutions is more efficient. We also ran a version of the LNS, in which the solution used at each iteration is not the incumbent, but the one provided by the previous iteration. This led to a worsening in solution costs of about 1%.

Table 5: Average results of LNS for the the four simulation strategies

		current	LNS - base demand				LNS - avg. demand				LNS - a priori demand			
month	#	cost	cost	gap	d_{un}	sec	cost	gap	d_{un}	sec	cost	gap	d_{un}	sec
Jan	5	6037.1	5653.0	-6.4	17.2	276.5	5603.6	-7.2	6.9	142.3	5329.5	-11.7	1.6	267.8
Feb	4	5718.8	5431.4	-5.0	16.3	275.0	5196.7	-9.1	5.1	281.1	5032.7	-12.0	1.4	256.9
Mar	4	6407.4	6142.9	-4.1	21.3	276.3	5712.3	-10.8	2.2	268.0	5538.2	-13.6	1.5	283.4
Apr	5	5426.2	5046.7	-7.0	10.0	272.4	5134.5	-5.4	3.9	281.3	4927.5	-9.2	1.3	268.5
May	4	6917.3	6501.1	-6.0	27.3	279.0	5941.6	-14.1	5.4	291.7	5706.7	-17.5	1.6	278.7
Jun	4	7246.9	6873.9	-5.1	32.1	279.5	6550.8	-9.6	12.9	277.4	6064.9	-16.3	1.7	286.4
Jul	5	6847.1	6508.1	-5.0	26.7	283.7	6133.6	-10.4	3.6	281.4	5912.9	-13.6	1.3	267.5
Aug	4	5176.7	4744.9	-8.3	9.6	272.9	4881.8	-5.7	6.1	277.2	4656.8	-10.0	1.0	278.7
Sep	4	7501.7	7163.7	-4.5	35.9	275.5	6577.4	-12.3	9.7	231.6	6111.1	-18.5	2.3	234.5
Oct	5	7931.7	7609.2	-4.1	41.2	279.9	6775.8	-14.6	8.0	52.0	6572.4	-17.1	5.8	198.3
Nov	4	9033.0	8774.9	-2.9	56.6	280.6	7571.5	-16.2	13.7	212.0	7176.0	-20.6	8.7	149.9
Dec	4	6984.1	6752.7	-3.3	37.4	269.2	6152.8	-11.9	13.8	281.6	5647.1	-19.1	4.7	175.7
avg		6769.0	6433.5	-5.1	27.6	276.7	6019.4	-10.6	7.6	239.8	5723.0	-14.9	2.7	245.5

Clearly, *a priori demand* cannot be applied in practice, but *avg demand* seems a good compromise. It models fluctuations on the demand and allows for cost savings of about 10%. Figure 5 provides a graphical visualization of the results in Table 5 and substantiates

these conclusions.

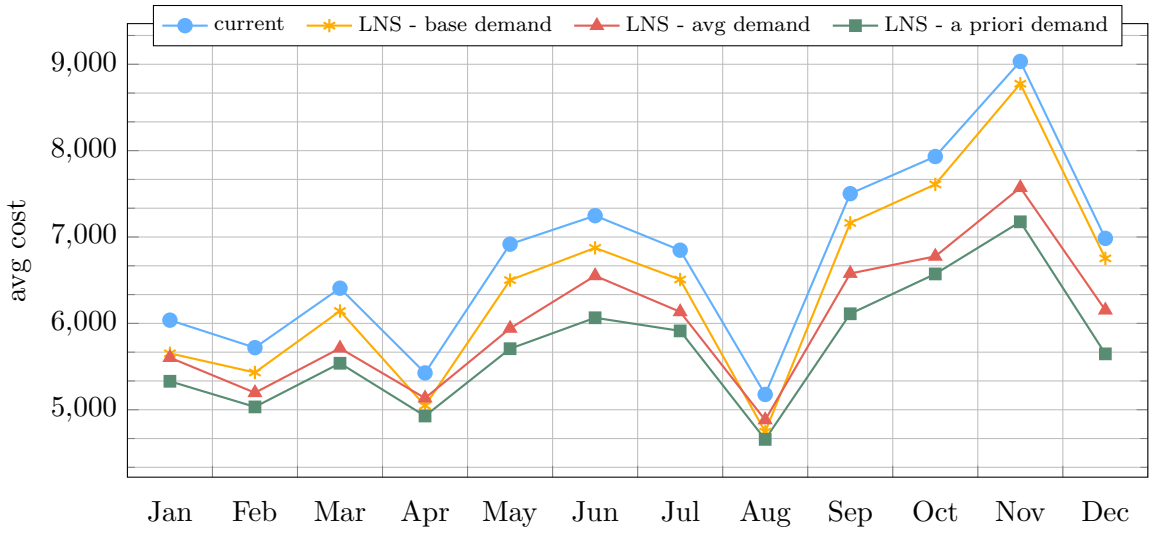


Figure 5: Average results of LNS for the four simulation strategies in a chart

6.3. Evaluation of the quality of service constraints

In Table 6, we present the results of an analysis on the impact of imposing a certain quality of service (QoS) for the design of time slot tables. In this case, formulation TIF^+ is used to both provide an initial solution and repair destroyed solutions, and we set $\alpha_m = 0.3$, $\alpha_a = 0.3$ and $gap = 2$. Thus, time slot tables produced by this approach, namely LNS_{QoS}, impose a minimum of 30% of resources allocated to morning time slots, another 30% to afternoon slots, and also guarantee that there cannot be more than 2 consecutive days without any offer of time slots in any region. The first column of each approach refers to the expected average monthly cost of solutions, while the second column of LNS_{QoS} specifies the average increase in the solution cost with respect to the LNS on the same scenario.

As expected, these results indicate that imposing QoS constraints has a negative impact on the solution cost, but it is quite limited. The average increase in cost is about 3.5% and it is stable over the three scenarios. Despite the additional requirements, our LNS still achieves significant improvements over the time slot tables used by the company. In the first scenario, LNS_{QoS} shows 2.1% of average improvements, while in the second and third, these values reach 8% and 11.2%, respectively.

In general, although QoS constraints might not be profitable, they are very important to provide customers with a more meaningful set of options and improve the overall satis-

Table 6: Impact of imposing QoS constraints on the expected solution cost

		base demand			avg demand			a priori demand		
		LNS	LNS_{QoS}		LNS	LNS_{QoS}		LNS	LNS_{QoS}	
month	#	cost	cost	gap	cost	cost	gap	cost	cost	gap
Jan	5	5653.0	5880.8	4.0%	5603.6	5774.7	3.1%	5329.5	5608.5	5.2%
Feb	4	5431.4	5636.5	3.8%	5196.7	5273.2	1.5%	5032.7	5199.8	3.3%
Mar	4	6142.9	6328.0	3.0%	5712.3	5944.2	4.1%	5538.2	5786.7	4.5%
Apr	5	5046.7	5241.6	3.9%	5134.5	5254.6	2.3%	4927.5	5130.0	4.1%
May	4	6501.1	6734.5	3.6%	5941.6	6208.6	4.5%	5706.7	5990.8	5.0%
Jun	4	6873.9	7024.9	2.2%	6550.8	6784.2	3.6%	6064.9	6382.3	5.2%
Jul	5	6508.1	6733.2	3.5%	6133.6	6314.3	2.9%	5912.9	6244.6	5.6%
Aug	4	4744.9	4980.0	5.0%	4881.8	5009.5	2.6%	4656.8	4852.1	4.2%
Sep	4	7163.7	7380.5	3.0%	6577.4	6743.6	2.5%	6111.1	6374.7	4.3%
Oct	5	7609.2	7817.1	2.7%	6775.8	6995.6	3.2%	6572.4	6815.6	3.7%
Nov	4	8774.9	8947.4	2.0%	7571.5	7691.1	1.6%	7176.0	7489.6	4.4%
Dec	4	6752.7	6866.2	1.7%	6152.8	6386.6	3.8%	5647.1	5793.2	2.6%
avg		6433.5	6630.9	3.2%	6019.4	6198.3	3.0%	5723.0	5972.3	4.3%

faction. This is particularly interesting to companies that have to compete with others to get public contracts, and where every feature that improves the quality of service counts.

7. Conclusions and future research directions

We have studied a practical attended home delivery problem arising in the context of an Italian service provider. Since 2012, this company provides an online application in which customers can request services by choosing a time slot of their liking.

The territory of action is divided into 12 regions, each having a dedicated time slot table that varies according to the associated expected demand. Currently, these time slot tables are manually built based on historical information of service performance, and changes are implemented as needed. Developing an automated approach to produce cost efficient time slot tables involves a complex two-stage stochastic decision problem. Time slot tables have to be generated considering a certain expected demand, then customer choices are simulated and an assignment plan is designed. Lastly, routes for each available technician are designed to minimize routing costs. Only then, the quality of the time slot tables can be assessed. The main objective of this paper was to propose an effective approach that allows for the creation of time slot tables that are cost efficient and adapt to the varying demand along the year.

One of the critical aspects of this problem is the evaluation of the expected cost of a solution, given that demand is not known beforehand and it is a key component to define

the routing costs. We proposed an LNS algorithm that uses as a repair method an ILP model to produce feasible solutions, while optimizing the distribution of resources in such a way that good features are preserved. Then, we simulate customer choices of time slots using one of four simulation strategies. Having the assignment plan, a second ILP model is used to estimate the associated routing costs.

We performed computational experiments on three different demand scenarios. In the first, we considered the same expected demand as in the time slot tables implemented at the company. This is the most restrictive scenario because it completely undermines demand fluctuations that are common in practice. The second scenario considers instead the average monthly demand to propose more flexible and adaptable time slot tables, while the third considers a utopic case where demand is known, but not customer choices. Results indicate that average improvements of about 5%, 10% and 15% can be achieved in each scenario, respectively.

As a future research direction, it would be interesting to study a rolling horizon approach to explore the dynamic aspects of the problem and evaluate the trade-offs of postponing the delivery of some services instead of resorting to outsourcing. Another potential development is to evaluate the impact of changing the arrangement of regions, which can lead to further improvements. We finally mention that the use of a demand forecast system could potentially increase the savings obtained by our optimization approach.

Acknowledgments

The authors would like to thank IRETI for the opportunity to work on this problem. Manuel Iori and Bruno Bruck also acknowledge financial support by Capes (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) under grant PVE no. A007/2013. We are also grateful to two anonymous referees for making valuable suggestions that helped improve the quality of the paper.

References

N. Agatz, A.M. Campbell, M. Fleischmann, and M. Savelsbergh. Challenges and Opportunities in Attended Home Delivery. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 379–396. Springer US, Boston, MA, 2008.

N. Agatz, A. Campbell, M. Fleischmann, and M. Savelsbergh. Time Slot Management in Attended Home Delivery. *Transportation Science*, 45(3):435–449, aug 2011.

R. Baldacci, A. Mingozzi, and R. Roberti. Recent Exact Algorithms for Solving the Vehicle Routing Problem Under Capacity and Time Window Constraints. *European Journal of Operational Research*, 218(1):1–6, 2012.

T. Bektas. The Multiple Traveling Salesman Problem: An Overview of Formulations and Solution Procedures. *Omega*, 34(3):209–219, 2006.

O. Bräysy and M. Gendreau. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, 39(1):104–118, 2005a.

O. Bräysy and M. Gendreau. Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005b.

A. Campbell and M. Savelsbergh. Decision Support for Consumer Direct Grocery Initiatives. *Transportation Science*, 39(3):313–327, 2005.

A. Campbell and M. Savelsbergh. Incentive Schemes for Attended Home Delivery Services. *Transportation Science*, 40(3):327–341, aug 2006.

X. Chen, B.W. Thomas, and M. Hewitt. The technician routing problem with experience-based service times. *Omega*, 61:49–61, 2016.

C. Cleophas and J. F. Ehmke. When are deliveries profitable. *Business and Information Systems Engineering*, 6(3):153–163, 2014.

G. Desaulniers, J. Lavigne, and S. Soumis. Multi-depot vehicle scheduling problems with time windows and waiting costs. *European Journal of Operational Research*, 111(3):479–494, 1998.

European Parliament. Directive 2009/73/EC of the European parliament and of the council, 2009. Available at http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:211:0094:0136:en:PDF.

C. Groër, B. Golden, and E. Wasil. The Consistent Vehicle Routing Problem. *Manufacturing & Service Operations Management*, 11(4):630–643, 2009.

I. Kara and T. Bektas. Integer Linear Programming Formulations of Multiple Salesman Problems and its Variations. *European Journal of Operational Research*, 174(3):1449–1458, 2006.

R. Liu, X. Xie, and T. Garaix. Hybridization of tabu search with feasible and infeasible local searches for periodic home health care logistics. *Omega*, 47:17–32, 2014.

D. Pisinger and S. Ropke. Large Neighborhood Search. In *Handbook of Metaheuristics*, pages 399–419. Springer, 2010.

M. Punakivi and J. Saranen. Identifying the Success Factors in e-Grocery Home Delivery. *International Journal of Retail & Distribution Management*, 29(4):156–163, 2001.

P. Shaw. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In Michael Maher and Jean-Francois Puget, editors, *Principles and Practice of Constraint Programming CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin Heidelberg, 1998.

R. Spliet and G. Desaulniers. The Discrete Time Window Assignment Vehicle Routing Problem. *European Journal of Operational Research*, 244(2):379–391, 2015.

R. Spliet and A. F. Gabor. The Time Window Assignment Vehicle Routing Problem. *Transportation Science*, 49(4):721–731, nov 2015.

X. Yang, A.K. Strauss, C.S.M. Currie, and R. Eglese. Choice-Based Demand Management and Vehicle Routing in E-Fulfillment. *Transportation Science*, 50(2):473–488, 2016.