UNIVERSITY OF MODENA AND REGGIO EMILIA

PH.D THESIS

Doctoral Course in Labour, Development and Innovation

XXXV cycle

SCHOOL OF DOCTORATE E4E

(Engineering for Economics - Economics for Engineering)

Department of Economics Marco Biagi

Marco Biagi Foundation

Design and development of optimization methods to support decisions in distributed logistics systems

---

PhD candidate:
Dott. Giorgio Zucchi

Advisor:
Prof. Manuel Iori
co-Advisor:
Prof. Carlo Alberto Magni
PhD coordinator:
Prof. Tindara Addabbo

2021/2022

UNIVERSITY OF MODENA AND REGGIO EMILIA

Corso di Dottorato in Lavoro, sviluppo ed innovazione
Dipartimento di economia Marco Biagi - Fondazione Marco Biagi

Doctor of Philosophy

Design and development of optimization methods to support decisions in distributed logistics systems

by Giorgio Zucchi

Abstract

In the last years, due to technological progress, optimization methods based on mathematical programming and heuristic algorithms have consistently improved their efficiency. It is now possible to obtain, in a large variety of problems, optimal or sub-optimal solutions in acceptable computational time. Emerging trends, driven by Industry 4.0 and Big Data revolutions, are pushing to combine optimization, at its purest mathematical level, with data science and decision support systems. The research conducted in this PhD thesis investigates the creation of decision support systems for the selection of optimal or sub-optimal solutions to optimize distributes services. The research presents the use of mathematical models and heuristic techniques to solve issues related to relevant problems, such as Personnel Scheduling and Vehicle Routing. The addressed case studies are derived from real business cases from Coopservice scpa, a large provider of services in relevant sectors, including healthcare and industry.

Introduction

In the last years, due to technological progress, the Optimization Methods (OM) based on mathematical programming and optimization algorithms have improved their efficiency and it is now possible to obtain, in a large variety of problems, optimal or sub-optimal solutions in acceptable CPU time. However, OMs individually could be fragile when changes in input conditions or into established constraints occur, therefore a great effort is required to use them in an operational context. Emerging trends, driven by industry 4.0 and Big Data revolutions, are pushing to combine optimization, at its purest mathematical level, with Data Science (DS) and Decision Support Systems (DSS). The increasing power of Personal Computers, the development of larger networks and databases, and the consequent application of mathematical models and methods represent the technological developments that stimulate the interest, on the part of companies, in the use of an algorithm as a Decision Support System to reduce the risk of uncertainty in making decisions (Averweg, 2010).

Nowadays large companies collect a substantial amount of data. These data are not often used in their potential, but only stored and archived. Without a system of models and methods that allow the analysis and processing, it is essentially impossible for a strategic decision-maker to make the best decision in the best possible conditions. In complex strategic decision-making situations, decisions often have to be made within a finite set of feasible alternatives that meet multiple conflicting objectives (Belton and Stewart, 2002).

Using a DSS is different from the traditional Decision Maker approach, as the Decision Makers must know and understand the DSS that has been developed for them to take into consideration the advantages and limitations of the system. On the other hand, those who design the DSS must know and understand what functions and data need to be integrated into the system and how to do it. Due to this complexity, two levels of knowledge are needed to develop models and methods for decision support: it is important to have knowledge of the approach and the strategy of Decision Makers to build an effective DSS.

As mentioned above, for years, large companies have begun to archive all the data that can be collected from a Big Data perspective, especially oriented towards the application of Industry 4.0. As a result of this availability, a Decision Maker may find herself displaced and unable to analyze or understand it; consequently, it is necessary to create systems to synthesize them (Ishizaka and Nemery, 2013); furthermore, after making a decision in conditions of uncertainty, a person can discover, by learning from the relevant results, that another alternative would have been preferable (Bell, 1982).

The integration with the traditional approach of Operations Research is that the DSS does not aim to provide an optimal or definitive solution. The DSS combines the use of models and methods, Data Analytics techniques, and traditional data processing functions, allowing the use of this system to people who do not necessarily have a mathematical, computer, or programming basis. These systems, to be fully and robustly developed, need algorithms that must be tested on real cases and with a large database. This is why in recent years it is possible to develop ever more effective DSS. Examples can be found in (Bruck et al., 2017) and in (Alonso et al., 2019).

The application of DSS, therefore, can reduce the risk of uncertainty of the decision-maker for what concerns the economic feasibility of a project, the budgeting of an offer, and the technical design. In addition, it is possible to improve business processes. Today, in order to support the Digital Transformation of a company and to guarantee a competitive advantage, it is necessary that decisions are not made merely based on some empirical and experiential observations; conversely, they should exploit digital transformations and the employment of Industry 4.0, so as to get an idea of the bigger picture.

In this thesis, we focus on the development of models and methods regarding Decision Support Systems, by using a mathematical approach and heuristic algorithms. The aim is to

reduce the time necessary for the generation of solutions and to optimize costs. In the past, solutions were normally generated manually and took longer time for this reason. The case studies presented are all derived from real business cases to create data-driven algorithms.

The remainder of the thesis is organized as follows. Chapter 1 presents a combined algorithm based on an Iterated Local Search and a mathematical model to solve the Time Window Assignment Vehicle Routing Problem (this work has been published in Martins et al., 2021). Chapter 2 presents a mathematical model to solve a scheduling problem during the Covid-19 pandemic (this work has been published as Zucchi, Iori, and Subramanian, 2021). Chapter 3 addresses a real-life task and personnel scheduling problem arising in Coopservice to provide cleaning services inside a hospital (the work has been published as Campana et al., 2021, and won the best paper award at the International Conference on Enterprise Information Systems (ICEIS). In Chapter 4, the reader can find a case study regarding lead time prediction in the pharmaceutical logistics sector of the company (published as Oliveira et al., 2021). In Chapter 5, a real-life multi-period orienteering problem related to the activity of patrolling a vast area in order to provide security services is discussed. Preliminary results were presented as Zucchi et al., 2022 and the full version of the paper is now submitted for publication to Networks, an international journal.

Contents

List of Figures

List of Tables

No regrets, no matter what

Chapter 1

On solving the time window assignment vehicle routing problem[1]

In this chapter we propose a combined algorithm based on an Iterated Local Search (ILS) and a mathematical model to solve the Time Window Assignment Vehicle Routing Problem (TWAVRP). The TWAVRP appears when the volume of customer demands is uncertain and time windows should be allocated to customers so as to minimize expected travel costs. Our goal is to find a heuristic strategy that can efficiently improve the current TWAVRP solution methods in the literature. For this purpose, we first use an ILS algorithm to generate feasible sets of routes. Then, we invoke a Mixed Integer Linear Programming formulation that assigns time windows to customers and selects the subset of routes of minimum expected cost. Computational results performed on benchmark instances show that our algorithm is competitive with respect to the literature, especially for instances with more than 45 clients.

1.1 Introduction

Vehicle routing is a class of problems that appears in several combinatorial optimization studies due to their practical relevance, mainly in the areas of retail and transport (Toth and Vigo, 2014). The classical *Vehicle Routing Problem* (VRP) calls for shipping freight to customers located along a distribution network by means of a fleet of capacitated vehicles, with the aim of minimizing the delivery costs. Since its introduction in the 1950s, several variations of the VRP have become prominent in the literature. Among these, we mention the *Vehicle Routing Problem with Time Windows* (VRPTW), where a given interval of time in which deliveries should occur is associated with each customer.

Inspired by retail distribution networks, Spliet and Gabor, 2014 introduced the Time Window Assignment Vehicle Routing Problem (TWAVRP). The TWAVRP appears when the volume of customer demands is uncertain and time windows should be allocated to the customers located along a distribution network, so as to minimize the expected travel costs. In the TWAVRP, each endogenous time window, that has a fixed-width, must be associated within the exogenous time window of the client. The exogenous time windows is represented by the arrival and departure limits of a client. According to Neves-Moreira et al., 2018, the TWAVRP can be defined as a *two-stage stochastic optimization problem*. Given a set of customers to be visited within a regular period, the first stage decisions are to assign a set of time windows to customers, before demand is known. In the second stage, after requests are revealed for each day, delivery schedules respecting the assigned time windows must be designed.

[1]This work has been published as: Martins, L. B., Iori, M., Moreira, M. C. O. & Zucchi, G. (2021). "On Solving the Time Window Assignment Vehicle Routing Problem via Iterated Local Search". In Graphs and Combinatorial Optimization: from Theory to Applications, CTW2020 Proceedings, pp. 223–235.

The TWAVRP faced in this work is part of a research whose focus is to give an efficient and accurate solution for a routing problem faced by an Italian company providing logistics services in several distribution fields. One of the characteristics presented in the particular routing problem faced by the company is the presence of a time window assignment decision phase. Our purpose is to help the company to minimize the actual delivery time and the total cost of the service they offer. We decided to start our research by first looking at the combinatorial aspect of the TWAVRP, with the aim of focusing later on its application to the company case study. In particular, the main contribution of this paper is to provide an answer to the following question: "Is there a heuristic strategy that can efficiently solve the TWAVRP as defined by Dalmeijer and Spliet, 2018; Spliet and Gabor, 2014?". For this purpose, we propose an algorithm that generates a set of routes by invoking an *Iterated Local Search* (ILS) metaheuristic, and then selects the most appropriate routes through an auxiliary mathematical formulation.

The remainder of the paper is structured as follows. Section 1.2 presents a brief literature review concerning the VRPTW and the TWAVRP. Section 1.3 formally describes the TWAVRP and presents a mathematical model. Section 1.4 reports the methodology that we developed to solve the TWAVRP. Section 1.5 shows the computational experiments that we performed. Finally, Section 1.6 presents some conclusions and some future research perspectives.

1.2 Brief Literature Review

Due to the academic interest in VRP variants and to the need of solving difficult real-world problems, researchers have been focusing more and more on realistic VRP, studying the class of so-called *Rich Vehicle Routing Problems* (RVRP). The RVRP class deals with realistic objective functions, uncertainty, dynamism, and a variety of practical constraints related to time, distance, heterogeneous fleet, inventory and scheduling problems, to mention a few. The reader is referred to (Caceres-Cruz et al., 2014; Vidal et al., 2013; Vidal et al., 2014) for more details about RVRP variants.

In this work, we deal with the TWAVRP, a problem that has characteristics resembling the VRPTW. The VRPTW is a generalization of the VRP involving appropriate time intervals for performing services, called time windows. In this class of problems, customer service can only be started within the time window defined by the customer (Desrochers, Desrosiers, and Solomon, 1992). Recently, several interesting VRPTW applications have been addressed in the literature. Among these, we mention the delivery of food (Amorim et al., 2014), the electric vehicle recharging problem (Keskin and Çatay, 2018), and the use of anticipated deliveries in pharmaceutical distribution (Kramer, Cordeau, and Iori, 2019).

In literature, we found some examples of classical approaches of VRPTW that consider Branch-and-Price (Azi, Gendreau, and Potvin, 2010; Desrochers, Desrosiers, and Solomon, 1992) and Tabu Search algorithms (Ceschia, Gaspero, and Schaerf, 2011; Cordeau, Laporte, and Mercier, 2001), and also more recent studies with applications at delivery food (Amorim et al., 2014) and electric vehicles recharging problem (Keskin and Çatay, 2018).

The TWAVRP has characteristics that make it even harder to solve in practice than the VRPTW (Spliet, Dabia, and Woensel, 2018). Several methods have been developed for its solution in recent years. The problem was formally introduced by Spliet and Gabor, 2014. They considered a finite number of scenarios with given realization probabilities. They proposed a mathematical model involving a large number of variables, and solved it by a branch-cut-and-price algorithm. Computational experiments on 40 instances involving 10, 15, 20, and 25 customers proved the efficiency of the proposed algorithm.

Moreover, Spliet and Desaulniers, 2015 tackled a variant of the TWAVRP where, for each customer, a time window is selected from a set of possibilities. To solve the problem, they implemented a Branch-Price-and-Cut and a Tabu Search. The results they obtained on a new set of instances showed that an approach considering five scenarios led to an average cost reduction of about 3.6% compared to a single-scenario approach.

In their paper, Dalmeijer and Spliet, 2018 addressed the TWAVRP through a Branch-and-Cut algorithm. They considered branching strategies based on a set of precedence inequalities. The effectiveness of the algorithm was demonstrated through numerical experiments and comparisons with the literature.

Inspired by an European food retailer, Neves-Moreira et al., 2018 applied the TWAVRP to a real food distribution case study, involving around 200 customers, with time windows defined according to the product segments. Their problem considers both traveled distances and fleet requirements costs in the objective function. Their solution method uses three phases: route generation; initial solution construction; and improvement by a matheuristic.

Finally, Spliet, Dabia, and Woensel, 2018 proposed a mathematical formulation for a TWAVRP variant that includes time-dependent travel times. To deal with this new problem, they applied a Branch-Price-and-Cut algorithm. Computational tests were run on artificial instances having up to 25 customers. The best solution value they found was only 0.55% higher, on average, than the optimal solution value.

1.3 Formal Problem Definition

We deal with the same problem described by Dalmeijer and Spliet, 2018. Consider a set of clients denoted by $H = \{1,2,\ldots,n\}$. A graph $G = (N,A)$ models the network of this problem, where $N = H \cup \{0,n+1\}$ is the overall set of nodes and 0 and $n+1$ represent, respectively, the departure and arrival depot nodes of all routes. A set A_H of arcs indicates the connections between any pair of customer nodes $i, j \in H$. Similar to set N, we denote $A = A_H \cup \{(0,j) \cup (j,n+1)$ for all $j \in H\}$, as the overall set of arcs connecting customers and depot nodes. Each arc $(i,j) \in A$ has an associated travel time t_{ij} and a travel cost c_{ij}. The travel times are non-negative and respect the triangular inequality ($t_{ij} \leq t_{ik} + t_{kj}$ for all i, j, and k), and the same applies to travel costs.

An unlimited set of homogeneous vehicles with capacity Q is available at the departure depot. We consider a set Ω of demand scenarios, each having probability of occurrence p_ω, for $\omega \in \Omega$, in such a way that $\sum_{\omega\in\Omega} p_\omega = 1$. Each customer $j \in H$ has a demand in scenario $\omega \in \Omega$ given by $0 \leq q_j^\omega \leq Q$.

Each client $j \in H$ has to be assigned to an endogenous time window of width w_j, which must be selected in a fixed exogenous time window $[e_j, l_j]$ provided in input, where $l_j - e_j \geq w_j$. A time window $[e_0, l_0]$ represents the opening hours of the departure depot. Similarly, a time window $[e_{n+1}, l_{n+1}]$ represents the opening hours of the arrival depot. The objective function consists in minimizing the expected traveled cost over all scenarios, which is given by $\min \sum_{\omega\in\Omega} p_\omega \sum_{(i,j)\in A} c_{ij} x_{ij}^\omega$, where x_{ij}^ω is a binary variable that takes the value 1 if arc $(i,j) \in A$ is traveled in scenario ω, 0 otherwise. For each scenario $\omega \in \Omega$, a route is feasible if the exogenous time window constraints are satisfied, the capacity constraints are satisfied and the customer j must be visited after the service time at customer i added to the travel time t_{ij} in the case that the customer j is visited after i.

1.4 An Iterated Local Search-based Algorithm

The heuristic method proposed in this paper is outlined in Algorithm 1. It is based on two successive phases, the first used to generate routes and the second used to select a subset of

routes having minimum cost. A similar idea was adopted by Moreira and Costa, 2013, who efficiently solved a quite different combinatorial optimization problem involving job rotation schedules in assembly lines with heterogeneous workers. Our method is composed of two parts. First, we generate a pool of feasible routes, minimizing the total cost of each scenario (Lines $3-5$), subject to vehicle capacity constraints and exogenous time windows. Then, we call an auxiliary *Mixed Integer Linear Programming* (MILP) formulation to select the most appropriate routes of the set, so as to optimize the total cost over all scenarios (Line 6) by respecting the generated endogenous time windows. The reference framework of Phase 1 is the ILS introduced by Lourenço, Martin, and Stützle, 2003. Such ILS has four components: (i) initial solution generator; (ii) local search procedure; (iii) perturbation; and (iv) acceptance criterion. The choice of this metaheuristic derives from the fact that it has been successfully applied in several combinatorial optimization problems (Avci and Topaloglu, 2017; Gunawan, Lau, and Lu, 2015a; Nogueira, Pinheiro, and Subramanian, 2018), including a number of VRP variants (Haddadene, Labadie, and Prodhon, 2016; Subramanian and Anjos Formiga Cabral, 2008). Moreover, it contains fewer parameters to be fine-tuned with respect to other metaheuristics. In Line 4, we represent the ILS by function $ILS(I_\omega, \alpha, n_{iter})$, which returns the set of routes obtained by the execution of the metaheuristic after receiving in input data I_ω, that is all the data from scenario ω. Note that parameter α corresponds to the perturbation factor, whereas n_{iter} gives the number of iterations without improvements. Next, we explain each component of the ILS and of the subsequent mathematical formulation used to select the final set of routes.

Algorithm 1 Main algorithm

```
1: procedure MAIN ALGORITHM(I:instance)
2:     P ← ∅                                          ▷ Empty pool of routes
3:     for ω ∈ Ω do
4:         P ← P ∪ ILS(I_ω, α, n_iter)                ▷ Generating the set of routes for each scenario
5:     end for
6:     s ← RSM(P, I)                                  ▷ Route Selector Model (RSM)
7:     return (s, f(s))                               ▷ solution, and its objective function
8: end procedure
```

1.4.1 Iterated Local Search (ILS)

Algorithm 2 gives the heuristic invoked to create the initial solution for the proposed ILS. The algorithm is inspired by the greedy strategy presented by Zhigalov, 2018. Let $\tilde{H}_\omega$ be the set of all customers in scenario $\omega \in \Omega$, that is, all customers demands in that scenario and let H_ω be the set of all available clients for a data set I on scenario ω. First, $\tilde{H}_\omega$ is sorted according to the earliest start time of the exogenous time window (i.e., e_i, for $i \in \tilde{H}_\omega$) of the customers (Line 3). The main loop consists of Lines 4–15, and terminates when all customers have been assigned. In each iteration, an empty route is opened (Line 5), and the highest priority customers (according to the sorting in Line 3) are appended to the route, one at a time, if such assignment respects vehicle capacity and time window constraints (Lines 8–11). Feasibility is checked by invoking the *infeasible*($\mathcal{R}$) procedure. If the current route is feasible, customer j is included in the route ($\mathcal{R}$) under construction and then removed from $\tilde{H}_\omega$ (Line 14).

The Local Search (LS) method is composed of six elementary neighborhoods:

N1 *Relocate intra-route*: change position of a customer in a route;

N2 *Swap intra-route*: swap two customer positions in a route;

Algorithm 2 Constructive Heuristic (CH)

1: **procedure** CONSTRUCTIVE HEURISTIC (CH)(I, H_ω)
2: $s \leftarrow \emptyset$
3: $\tilde{\mathcal{H}} \leftarrow sort(H_\omega)$ ▷ sort clients in non-descending order of earliest exogenous time window
4: **while** $\tilde{\mathcal{H}} \neq \emptyset$ **do**
5: $\mathcal{R} \leftarrow \emptyset$
6: **for** $j \in \tilde{\mathcal{H}}$ **do**
7: $\mathcal{R} \leftarrow \mathcal{R} \cup \{j\}$
8: **if** *infeasible*($\mathcal{R}$) = **true then**
9: $\mathcal{R} \leftarrow \mathcal{R} \backslash \{j\}$
10: **else**
11: $\tilde{\mathcal{H}} \leftarrow \tilde{\mathcal{H}} \backslash \{j\}$
12: **end if**
13: **end for**
14: $s \leftarrow s \cup \mathcal{R}$
15: **end while**
16: **return** s ▷ feasible solution
17: **end procedure**

N3 *2-opt*: invert a sequence of customers allocated to the same route;

N4 *Relocate inter-route*: relocate a customer to a different route in the same scenario;

N5 *Swap inter-route*: exchange two customers allocated in different routes, in the same scenario;

N6 *Cross inter-route*: split two routes at given points and exchange their remaining parts.

The LS method invokes the neighborhoods according to the procedure shown in Algorithm 3. Given a solution s, a list $NL(s)$ of neighborhoods is initialized according to the inter-route neighborhoods (N4, N5, and N6). If $s^{'}$ is feasible and the distance performed, represented by function $f(s^{'})$, decreases compared to the current solution (Line 6), an intra-route search procedure (N1, N2, and N3) is performed over s'. If the intra-route procedure improves s', the current solution $\tilde{s}$ is used to replace s^* (Line 10). The process terminates when no inter-neighborhood can return an improvement.

Starting from a solution s^*, the Perturbation method invokes a list $NL(s^*)$ of possible neighborhood moves according to the inter-route neighborhoods (N4, N5, and N6). A percentage α of neighborhoods in $NI(s^*)$ is randomly chosen and applied to s^*. Regarding the Acceptance criterion, we accept only solutions that are better than the current one. Algorithm 4 summarizes the ILS that is applied to each scenario of Phase 1.

1.4.2 Route Selector Model

The ILS algorithm generates a set P of viable routes for each scenario $\omega \in \Omega$ (see Algorithm 1). Note that all routes in P respect the capacity and time-windows constraints. We built a MILP formulation, called *Route Selector Model* (RSM), whose aim is to choose the most appropriate subset of routes from P, assigning an endogenous time window to each client, over all scenarios.

To present the RSM, we take from P: (i) f_{jr}^{ω} as the starting time of service on client j on the route r in scenario ω; (ii) c_r^{ω} as the cost to choose a route $r \in P$ in scenario ω; and (iii) x_{jr}^{ω} as a binary parameter equal to one if client j belongs to route $r \in R_\omega$ in scenario

Algorithm 3 Local Search method (LS)

```
1:  procedure LOCAL SEARCH METHOD (LS)(s)
2:      s* ← s
3:      for N ∈ NL(s*) do                ▷ NL(s*): list of inter-neighborhoods of solution s*
4:          for s' ∈ N do
5:              if f(s') < f(s*) and feasible(s') = true then
6:                  s* ← s'
7:                  for N ∈ NI(s*) do    ▷ NI(s*): list of intra-neighborhoods of solution s'
8:                      for s̃ ∈ N do
9:                          if f(s̃) < f(s*) and feasible(s̃) = true then
10:                             s* ← s̃
11:                         end if
12:                     end for
13:                 end for
14:             end if
15:         end for
16:     end for
17:     return s*                        ▷ best feasible solution found
18: end procedure
```

ω, 0 otherwise. Consider u_r^ω as a binary variable equal to one if route $r \in P$ is selected, 0 otherwise, and y_i as a continuous variable that measures the starting time of the endogenous time window of customer $i \in \widetilde{H}_\omega$. Recall that, as indicated above, w_i gives the time window width of customer i. The RSM is as follows:

$$\min \sum_{\omega \in \Omega} p_\omega c_r^\omega u_r^\omega \tag{1.1}$$

subject to

$$\sum_{r \in R_\omega} x_{jr}^\omega u_r^\omega = 1 \qquad \forall j \in H, \omega \in \Omega \tag{1.2}$$

$$\sum_{r \in R_\omega} f_{jr}^\omega x_{jr}^\omega u_r^\omega \geq y_j \qquad \forall j \in H, \omega \in \Omega \tag{1.3}$$

$$\sum_{r \in R_\omega} f_{jr}^\omega x_{jr}^\omega u_r^\omega \leq y_j + w_i \qquad \forall j \in H, \omega \in \Omega \tag{1.4}$$

$$y_j \in [e_j, l_j - w_j] \qquad \forall j \in H, \omega \in \Omega \tag{1.5}$$

$$u_r^\omega \in \{0,1\} \qquad \forall \omega \in \Omega, r \in R_\omega. \tag{1.6}$$

The model optimizes the total cost of the selected routes. Constraints (1.2) indicate that each customer has to be served in all scenarios by a single route. Constraints (1.3)–(1.4) establish the endogenous time windows. Domain variables are presented by Constraints (1.5)–(1.6).

1.5 Computational Experiments

We performed a set of computational experiments aimed at assessing the performance of the ILS-based algorithm that we developed for the TWAVRP. The algorithms were implemented in *Python 3.7.4*, using the MILP solver *Gurobi 8.1.1* for the development of the RSM (Section 1.4.2), running a single thread for a time limit of 3600 seconds on each instance. All

Algorithm 4 Iterated Local Search (ILS)

procedure ITERATED LOCAL SEARCH (ILS)(H_ω, α, n_{iter})
  $s^* \leftarrow \emptyset$ ▷ Best solution found so far (take $f(s^*) = +\infty$)
  $s \leftarrow CH(H, H_\rightarrow)$ ▷ $H_\rightarrow$: set of available customers of data set H
  $s_{ls} \leftarrow LS(s)$
  $\mathcal{P} \leftarrow s_{ls} \cup s$ ▷ Initializing the set of feasible solutions
  $s^* \leftarrow s_{ls}$
  $count \leftarrow 0$
  while $count \neq n_{iter}$ **do**
    $s' \leftarrow Perturbation(s^*, \alpha)$
    $s_{ls} \leftarrow LS(s')$
    $\mathcal{P} \leftarrow \mathcal{P} \cup s' \cup s_{ls}$
    if $f(s') < f(s^*)$ **then**
      $s^* \leftarrow s'$
      $count \leftarrow 0$
    else
      $count \leftarrow count + 1$
    end if
  end while
  return $\mathcal{P}$ ▷ Set of feasible solutions found
end procedure

experiments were performed on a PC Intel i7, 3.5 GHz with 16 GB RAM, which is similar to the computer used by Dalmeijer and Spliet, 2018.

To generate the pool of routes, Algorithm 8 was executed five times on each instance. This number was tuned through preliminary tests in which we obtained a good trade-off between quality and computational effort. Furthermore, this value allowed the algorithm to make good use of its stochastic components. The number of iterations without improvements (n_{iter}) and the perturbation percentage (α) were fine-tuned through the *Irace* package (López-Ibáñez et al., 2016). For that purpose, we generated 200 training instances by using the instance generator proposed by Dalmeijer and Spliet, 2018. The values returned by the *Irace* package at the end of this test were n_{iter} = 100 and α = 0.35.

1.5.1 Instances

We use the set of TWAVRP instances proposed by Spliet, Dabia, and Woensel, 2018. Each instance considers a different combination of number of customers, vehicle capacity, demand for each scenario, probability of each scenario, size of exogenous and endogenous time windows, travel costs, and travel times. In this way, the data set comprises ninety instances divided into two classes: small instances and large ones. Small instances contain four sets of ten instances each, having 10, 15, 20, and 25 customers, respectively. Large instances contain five sets of ten instances each, with 30, 35, 40, 45, and 50 customers, respectively. The customer's coordinates were generated as uniformly distributed over a square with sides of length five. The depot is located in the center of the square. Each instance includes demands for each customer in three scenarios with equal probability of occurrence. Exogenous time windows are distributed as follows: a time window $[10, 16]$ is given to 10% of the customers; $[7, 21]$ to 30% of the customers; and $[8, 18]$ to the remaining 60%. The width of the endogenous time window is set to $w_i = 2$ for all customers. The costs and the travel times between the nodes were obtained by calculating the Euclidean distances between their coordinates.

TABLE 1.1: Average results aggregated by number of customers (10 instances per line, 5 ILS executions per instance)

Instance	CPU time (seconds)			Gaps	
N. customers	B&C	ILS	ILS+RSM	Gap*(%)	Gap(%)
10	0.1	4.50 ± 0.29	6.61 ± 0.56	0.34 ± 1.00	0.41 ± 1.02
15	4.5	16.50 ± 1.17	26.25 ± 1.86	0.00 ± 0.18	0.11 ± 0.25
20	2.2	39.06 ± 2.01	80.30 ± 7.49	0.02 ± 0.05	0.06 ± 0.10
25	12.4	68.48 ± 2.03	153.29 ± 18.56	0.06 ± 0.14	0.27 ± 0.78
30	544.0	107.27 ± 3.40	284.38 ± 12.62	0.04 ± 0.10	0.28 ± 0.39
35	1,531.7	161.59 ± 9.48	501.77 ± 97.94	0.02 ± 0.13	0.29 ± 0.42
40	3,252.0	224.33 ± 6.11	749.92 ± 41.11	0.10 ± 0.52	0.72 ± 0.73
45	3,600.0	289.34 ± 28.78	990.15 ± 172.79	-0.69 ± 0.83	-0.18 ± 1.61
50	3,600.0	372.98 ± 24.41	1,743.16 ± 261.71	-1.89 ± 0.12	-1.62 ± 1.31

1.5.2 Results

The experiments compare our ILS based-algorithm with the *Branch-and-Cut* (B&C) proposed by Dalmeijer and Spliet, 2018, which can be considered the state-of-art method for the solution of the TWAVRP. The results that we obtained are summarized in Table 1.1. They are aggregated for instances having the same quantity of clients (first column). The remaining columns contain the average and the standard deviation of each measure, with the exception of the B&C time as it was executed just once. Columns B&C, ILS, and ILS+RSM, under the group CPU time (seconds), give the computational times spent by, respectively, the algorithm by Dalmeijer and Spliet, 2018, the ILS for the construction of the pool of routes, and the overall Algorithm 1 including ILS and RSM. The columns named Gap*(%) and Gap(%) indicate the percentage deviation of the average solution value found over all repetitions with respect to the best solution value, and with respect to the best-known values obtained by the B&C method, respectively.

Regarding instances that have between 10 to 35 clients, we can observe that our method found relative average deviations from 1.5% to 0.16% compared with the B&C solutions in the worst case, and that the total time of the five executions of ILS+RSM is higher than the B&C execution time. In the group of larger instances (45- 50 clients), the ILS+RSM outperforms the results found in literature concerning both best-found and average solution values of the five performed tests.

Table 1.2 highlights the behavior of our method on the 20 larger instances having 45 and 50 customers. We report the lower bound and upper bound obtained in Dalmeijer and Spliet, 2018 (columns LB and UB, respectively), and the best (column Best) and average (column Avg) solution values found by our ILS+RSM method. In the problems with 45 clients, both methods were competitive, each finding the best results for about half of the cases. Our method improved the solution cost obtained by the B&C for all instances with 50 clients, both considering columns Best and Avg. We estimate that the diversity of routes caused by different local search operators was beneficial for the performance of the ILS+RSM algorithm for these most difficult instances. Overall, we can conclude that the ILS+RSM is a good heuristic method for moderate and large size instances of the TWAVRP. Our research will now focus on adapting it to the real-world case study that motivated our study, so as to embed possible complicating constraints and solve even larger instances.

TABLE 1.2: Results for instances with 45-50 customers (best UB values appear in bold)

Instance		B&C by Dalmeijer and Spliet, 2018		ILS + RSM	
#	N. customers	LB	UB	Best UB	Avg UB
71	45	49.52	51.78	**51.22**	**51.41**
72	45	50.73	**52.13**	51.86	52.94
73	45	41.50	**41.70**	41.95	42.24
74	45	47.25	**47.84**	47.96	48.16
75	45	48.77	49.86	**49.47**	50.02
76	45	48.38	52.09	**49.90**	**50.03**
77	45	50.09	**51.18**	**51.18**	51.25
78	45	52.02	53.95	**53.35**	**53.74**
79	45	47.45	**48.21**	48.27	48.69
80	45	49.57	**50.57**	50.61	50.78
81	50	56.81	58.85	**58.16**	**58.29**
82	50	51.50	53.20	**52.98**	**53.03**
83	50	57.45	60.67	**58.77**	**58.89**
84	50	52.31	56.38	**54.09**	**54.23**
85	50	53.74	56.07	**55.06**	**55.26**
86	50	51.68	54.76	**53.02**	**53.16**
87	50	52.47	54.14	**53.81**	**53.87**
88	50	54.82	56.91	**56.27**	**56.36**
89	50	59.23	61.51	**60.32**	**60.62**
90	50	57.68	59.55	**58.95**	**59.23**

1.6 Conclusions and Future Research Avenues

We studied the *Time Windows Assignment Vehicle Routing Problem* (TWAVRP), a VRP variant that appears when the volume of customer demands is uncertain and visits over multiple days should be planned. The objective is to create routes that minimize expected travel costs, assigning a time window over all scenarios to each customer, and respecting the vehicle capacity. Our interest in this problem derives from a real-world case study. We decided to begin our research with the development of a good and flexible metaheuristic, and to test it on the benchmark TWAVRP instances, so as to check if good-quality solutions can be found within reasonable computational efforts.

To this aim, we proposed an *Iterated Local Search* (ILS) algorithm that generates a pool of feasible routes for each scenario, and a mathematical model, called *Route Selector Model* (RSM), that chooses the most appropriate routes, among those created, in order to minimize total costs and indicate the time windows for the customers. We compared the results of our algorithm (ILS+RSM) with the Branch-and-Cut proposed by Dalmeijer and Spliet, 2018. The ILS+RSM presented competitive results, concerning both solution quality and computational effort, in particular for the larger size instances involving 45 and 50 customers.

Interesting avenues of further research concern: (i) incorporating new complicating constraints deriving from the real-world case study in the metaheuristic; (ii) testing other neighborhood-based metaheuristics as generators of routes; (iii) testing multiple calls to the RSM with different pools of routes. This last avenue is motivated by the fact that in our tests the RSM converged quickly to the incumbent solution, so there is hope to find good solution values by invoking it multiple times.

Chapter 2

Personnel scheduling during Covid-19 pandemic[1]

This chapter addresses a real-life personnel scheduling problem in the context of Covid-19 pandemic, arising in Coopservice. In this case study, the challenge is to determine a schedule that attempts to meet the contractual working time of the employees, considering the fact that they must be divided into mutually exclusive groups to reduce the risk of contagion. To solve the problem, we propose a mixed integer linear programming formulation (MILP). The solution obtained indicates that optimal schedule attained by our model is better than the one generated by the company. In addition, we performed tests on random instances of larger size to evaluate the scalability of the formulation. In most cases, the results found using an open-source MILP solver suggest that high quality solutions can be achieved within an acceptable CPU time. We also project that our findings can be of general interest for other personnel scheduling problems, especially during emergency scenarios such as those related to Covid-19 pandemic.

2.1 Introduction

Personnel scheduling problems traditionally consist of optimizing work timetables, i.e., determining the appropriate times and shifts the employees of a company should work. Several objectives can be considered, such as minimizing duty costs, maximizing productivity, or minimizing the number of employees. The solutions generated must meet different criteria, for example, maximum number of working days, maximum amount of working hours, and so on. Due to the recent Covid-19 pandemic, we introduce a novel and emerging aspect: the risk of contagion among the workers. Covid-19 is now widely known to be highly contagious, and the interested reader may refer to, e.g., (Harapan et al., 2020; Li et al., 2020) and (Shereen et al., 2020) for further information regarding the transmission of the virus and the source of the disease, respectively.

On the one hand, there is a rich body of literature on how governments can avoid the spread of pandemic diseases (e.g., Aledort et al., 2007; Flahault et al., 2006; Fang, Nie, and Penny, 2020), or how physicians can deal with certain scenarios arising in medical centers, where they are faced with a growing, sometimes even uncertain, demand for emergency care (Marchesi, Hamacher, and Fleck, 2020; Tohidi, Kazemi Zanjani, and Contreras, 2020). On the other hand, the same does not happen when it comes to investigating optimized personnel scheduling policies that take into account the risk of disease spreading in specific environments. This is the case of companies that must remain open to provide essential services during a possible quarantine period. In this work, we attempt to start filling this gap by studying and solving a real-world personnel scheduling problem during Covid-19 pandemic, arising

[1]This work has been published as: Zucchi, G., Iori, M., & Subramanian, A. (2021). Personnel scheduling during Covid-19 pandemic. Optimization Letters, 15(4), 1385-1396.

in a large Italian pharmaceutical distribution warehouse operated by Coopservice (Coopservice Scpa, 2020). It consists of a large service company operating in several fields such as transportation, logistics, cleaning and security services.

There is a vast literature on personnel scheduling and we refer to Ernst et al., 2004a; Ernst et al., 2004b; Van den Bergh et al., 2013 for detailed surveys and to Brucker, Qu, and Burke, 2011 for models and complexity results. According to the classification scheme presented in Ernst et al., 2004a, we can categorize the scheduling made by the company before and after Covid-19 pandemic as demand-based and shift-based, respectively. The latter is extensively adopted in nurse scheduling studies (e.g., El-Rifai, Garaix, and Xie, 2016; Ozkarahan and Bailey, 1988). In particular, a variant of the problem was tackled in Vanden Berghe, 2002, in which nurses with the same skills, or that are married to each other and with children, were not allowed to work together during the same shift. This situation somewhat resembles the one found in our problem (see Section 2.2), where an employee originally assigned to a sector can no longer be reassigned to another one during the workday, as it often happened before the pandemic. More recently, the study described in Seccia, 2020 proposed and solved a nurse rostering problem arising in Covid-19 pandemic, where the author considered an emergency scenario by allowing nurses to work in multiple shifts during a workday. The objective was to decrease the occurrences of shifts with an insufficient number of nurses, and to balance the schedules by minimizing the working hours of the nurses.

To solve the problem introduced in this work, we propose a mathematical formulation based on mixed integer linear programming (MILP). The results obtained show that our optimized solution is capable of improving the manual one produced by the company by 37.3%, while keeping the same number of workers. In addition, we demonstrate that the model is capable of obtaining, on average, high quality solutions on larger instances generated at random within an acceptable CPU time.

The remainder of the chapter is organized as follows. Section 2.2 formally introduces our personnel scheduling problem. Section 2.3 shows the solutions adopted by the company before and after Covid-19 pandemic, as well as the performance measures used to evaluate the solutions. Section 2.4 describes the proposed mathematical model. Section 2.5 contains the results of the computational experiments. Finally, Section 2.6 presents the concluding remarks.

2.2 Problem description

The activities that Coopservice need to perform are to receive, organize and distribute pharmaceutical products to hospitals. The company manages large regional warehouses in Italy, where goods are received, stored, picked-up, and finally delivered to the hospitals, according to their demand. Therefore, Coopservice is responsible for managing the last phase of the pharmaceutical supply chain: from the inbound activity to the last-mile delivery (Kramer, Cordeau, and Iori, 2019). The interest reader may refer to (Pazour and Meller, 2013) for an overview on how to manage the supply chain between a distribution center (as the one considered in this work) and a hospital pharmacy.

In this work, we study the case of a regional warehouse located in Tuscany, one of the largest regions in Italy. The warehouse is composed of two floors: the top floor stores all pharmacological items (e.g., antiblastic, fridge drugs, toxic and inflammable), whereas the ground floor the non-pharmacological ones (food, paper, plastic, masks, gloves, syringes, etc.). Each floor has two working sectors, thus leading to a total of four sectors. There are 23 employees working in the warehouse. In the following, we use E and A to define the sets of employees and sectors, respectively.

The arise of Covid-19 brought into light a new issue that affected the movement of workers in the warehouse. Before the pandemic, all of them could move from one sector to the other, when required, during the workday. This became no longer possible during the pandemic period, meaning that each employee $e \in E$ should work in exactly one sector $a \in A$ during the workday, with a view of avoiding contagion. As a result, the company was forced to modify the personnel scheduling policy by creating groups of employees that must always work together, both in the same shift and in the same sector.

In our problem, we also consider a set of shifts S where the employees can work. We denote by $s = 1$ and $s = 2$ the morning and afternoon shifts, respectively, from Monday to Friday. Because of the emergency scenario, employees must also work in sector $a = 1$ on Saturday morning, here referred to as shift $s = 3$. Each shift $s \in S$ has a maximum amount of weekly working hours for each sector $a \in A$, represented by λ_{sa}.

To balance the activities, each group of employees alternate their shift every week, i.e., if a group works on the morning shift during a week, in the next week the same group will have to work on the afternoon shift. Moreover, for each sector $a \in A$ and each shift $s \in S$, we define by τ_{sa} the weekly minimum amount of working hours imposed by the company to ensure the required productivity level. In addition, each employee $e \in E$ has a contractual amount of working hours h_e^{max} per week. Finally, we define c_{ea} as a binary input parameter indicating whether employee $e \in E$ can work in sector $a \in A$ ($c_{ea} = 1$) or not ($c_{ea} = 0$). This corresponds to the skill matrix of employees.

Let W be the set of weeks. The aim of this work is to solve the personnel scheduling problem for 2 consecutive weeks (i.e., $W = \{1,2\}$), satisfying the new shift regulation for the emergency period issued by the authorities. The objective of the problem is to minimize the sum of the deviations from the contractual amount of working hours of each worker formally given by the absolute difference between the weekly working time in the solution and h_e^{max}. Both positive and negative deviations imply in additional costs for the company (if negative, the missing hours need to be paid nevertheless to the worker, if positive, the extra hours impose an extra cost).

Table 2.1 shows the schedule manually built by the company to solve the problem for sector 1 before and after the pandemic. Note that the morning shift starts at 06:00 and finishes at 12:00, while the afternoon shift starts at 12:00 and finishes at 20:00. For each employee $e \in E$, we report the sum of weekly working hours $\sum h_e$ and the deviation Δ_{ew} in each $w \in W$ with respect to h_e^{max}. It can be seen that before the pandemic there was no deviation at all ($\Delta_{ew} = 0 \ \forall e \in E, w \in W$), whereas the same did not occur after the pandemic (where, e.g., employee 5 has a deviation of 10 working hours in week 2).

2.3 Personnel scheduling in a normal scenario and during Covid-19 pandemic

In this section, we analyze the schedules adopted by the company before and after the pandemic. Table 2.2 reports the weekly value of Δ_{ew} for each employee. One can clearly observe that the schedule determined by the company in a normal scenario successfully met the amount of contractual working hours for all employees but 11, 18 and 21. The new schedule generated by the company led, however, to many deviations, as highlighted in the bottom part of the table.

In addition to the evaluation of Δ_{ew}, we also need to evaluate the contagion risk. To this aim, Figure 2.1 depicts a network graph, before and after the pandemic, where the vertices denote the employees and the edges indicate whether a pair of employees worked together in the same shift and in the same sector (for at least one hour). From this graph representation, we introduced a contagion risk factor R to measure the risk of infection, based on the average

TABLE 2.1: Personnel scheduling for sector 1 before and after Covid-19 pandemic

Personnel scheduling for sector 1 before Covid-19 pandemic

Week 1

e	06-07	07-08	08-09	09-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	$s=3$	$\sum h_e$	h_e^{max}	Δ_{e1}
1																30	30	0
2																30	30	0
3																40	40	0
4																40	40	0
5																30	30	0
6																30	30	0

Week 2

e	06-07	07-08	08-09	09-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	$s=3$	$\sum h_e$	h_e^{max}	Δ_{e2}
1																30	30	0
2																30	30	0
3																40	40	0
4																40	40	0
5																30	30	0
6																30	30	0

Personnel scheduling for sector 1 after Covid-19 pandemic

Week 1

e	06-07	07-08	08-09	09-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	$s=3$	$\sum h_e$	h_e^{max}	Δ_{e1}
1															6 h	31	30	1
2																30	30	0
3																40	40	0
4																40	40	0
5																30	30	0
6																30	30	0

Week 2

e	06-07	07-08	08-09	09-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	$s=3$	$\sum h_e$	h_e^{max}	Δ_{e2}
1																30	30	0
2																30	30	0
3															6 h	36	40	4
4															6 h	36	40	4
5																40	30	10
6																40	30	10

vertex degree of the graph. Let G^{aw} be a graph associated with each sector $a \in A$ and each week $w \in W$. Every graph has a vertex-set $V(G^{aw})$ and edge-set $E(G^{aw})$. For each vertex $u \in V(G^{aw})$ we define $\Omega(u)$ as its corresponding degree (number of first neighbors). The risk factor R_{aw} for each sector a and for each week w is thus formally defined as:

$$R_{aw} = \frac{\sum_{u \in V(G^{aw})} \Omega(u)}{|V(G^{aw})|}. \tag{2.1}$$

We can then determine the value of the risk factor R_a for each sector a as follows:

$$R_a = \frac{\sum_{w \in W} R_{aw}}{|R_{aw}|}. \tag{2.2}$$

For example, when considering Sector 2, we can observe that the average vertex degree (i.e., the risk factor R) before and after the pandemic were 3 and 1, respectively.

Table 2.3 shows the average values of R for each sector before and after the pandemic. It can be seen that the new schedule visibly decreases the risk of disease spreading, as the values of R are considerably smaller, when compared to the previous situation.

2.4 Proposed mathematical formulation

Let x_{se}^{wa} be a binary variable assuming the value one if employee $e \in E$ is assigned to shift $s \in S$ to work in sector $a \in A$ in week $w \in W$, 0 otherwise. In addition, let y_{se}^{wa} be the amount of working hours spent in sector $a \in A$ by employee $e \in E$ assigned to shift $s \in S$ in week $w \in W$. Finally, the excess and lack of working hours associated with employee $e \in E$ in week $w \in W$ are defined by variables δ_{ew}^{+} and δ_{ew}^{-}, respectively. The MILP formulation can be written as follows.

TABLE 2.2: Weekly working hours before and after Covid-19 pandemic

Weekly working hours before Covid-19 pandemic											
Sector 1			Sector 2			Sector 3			Sector 4		
e	Δ_{e1}	Δ_{e2}	e	Δ_{e1}	Δ_{e2}	e	Δ_{e1}	Δ_{e2}	e	Δ_{e1}	Δ_{e2}
1	0	0	7	0	0	11	1	1	18	1	1
2	0	0	8	0	0	12	0	0	19	0	0
3	0	0	9	0	0	13	0	0	20	0	0
4	0	0	10	0	0	14	0	0	21	1	1
5	0	0				15	0	0	22	0	0
6	0	0				16	0	0	23	0	0
						17	0	0			

Weekly working hours after Covid-19 pandemic											
Sector 1			Sector 2			Sector 3			Sector 4		
e	Δ_{e1}	Δ_{e2}	e	Δ_{e1}	Δ_{e2}	e	Δ_{e1}	Δ_{e2}	e	Δ_{e1}	Δ_{e2}
1	1	0	7	10	0	11	1	4	18	6	1
2	0	0	8	0	10	12	0	0	19	0	5
3	0	4	9	0	10	13	5	0	20	5	0
4	0	4	10	0	10	14	5	0	21	4	1
5	0	10				15	5	0	22	5	0
6	0	10				16	0	5	23	0	5
						17	0	0			

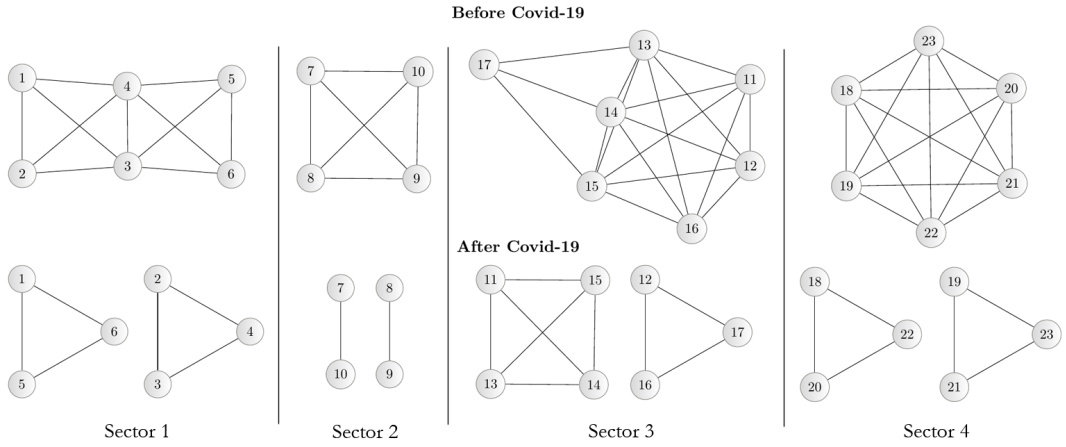


FIGURE 2.1: Network of relationships between employees

TABLE 2.3: R values before and after Covid-19 pandemic in a company solution

	Sector 1	Sector 2	Sector 3	Sector 4
R_{before}	3.7	3.0	5.7	5.0
R_{after}	2.0	1.0	2.5	2.0

$$\min \Delta: \sum_{e \in E} \sum_{w \in W} \delta_{ew}^{-} + \delta_{ew}^{+} \tag{2.3}$$

$$\text{s.t.} \quad \sum_{a\in A}\sum_{s\in\{1,2\}} x_{se}^{wa} = 1 \quad w\in W, e\in E \tag{2.4}$$

$$\sum_{s\in\{1,2\}} x_{se}^{1a} = \sum_{s\in\{1,2\}} x_{se}^{2a} \quad a\in A, e\in E \tag{2.5}$$

$$\sum_{a\in A}\sum_{w\in W} x_{se}^{wa} = 1 \quad e\in E, s\in\{1,2\} \tag{2.6}$$

$$x_{3e}^{w1} \leq x_{1e}^{w1} \quad e\in E, w\in W \tag{2.7}$$

$$x_{3e}^{wa} = 0 \quad w\in W, a\in\{2,3,4\}, e\in E \tag{2.8}$$

$$y_{se}^{wa} \leq \lambda_{sa} x_{se}^{wa} \quad w\in W, e\in E, a\in A, s\in S \tag{2.9}$$

$$\sum_{e\in E} y_{se}^{wa} \geq \tau_{sa} \quad w\in W, a\in A, s\in S \tag{2.10}$$

$$\sum_{a\in A}\sum_{s\in S} y_{se}^{wa} + \delta_{ew}^{-} - \delta_{ew}^{+} = h_e^{max} \quad w\in W, e\in E \tag{2.11}$$

$$x_{se}^{wa} \leq c_{ea} \quad w\in W, e\in E, a\in A, s\in S \tag{2.12}$$

$$x_{se}^{wa} \in \{0,1\} \quad w\in W, e\in E, a\in a, s\in S \tag{2.13}$$

$$y_{se}^{wa} \geq 0 \quad w\in W, e\in E, a\in A, s\in S \tag{2.14}$$

$$\delta_{ew}^{+}, \delta_{ew}^{-} \geq 0 \quad e\in E, w\in W. \tag{2.15}$$

Objective function (2.3) minimizes the total deviation Δ between the amount of weekly contractual hours for each worker (h_e^{max}) and the actual working hours (y_{se}^{wa}). Constraints (2.4) impose that each employee must be weekly assigned to exactly one shift $s\in\{1,2\}$ and one sector $a\in A$. Constraints (2.5) ensure that if an employee $e\in E$ is designated to sector $a\in A$ in the first week ($w=1$), he/she must work in the same sector in the next week ($w=2$). Constraints (2.6) impose each employee to work in exactly one sector per shift. Constraints (2.7) state that an employee can only work on the Saturday shift ($s=3$) if he/she is assigned to the morning shift $s=1$ to work in sector $a=1$. Constraints (2.8) forbid all employees to work in sectors 2, 3 and 4 on the Saturday shift. Constraints (2.9) enforce a maximum amount of weekly working hours, whereas constraints (2.10) ensure that the working hours requested for every shift in each sector are respected. Moreover, constraints (2.11) indicate the contractual amount of weekly working hours for each employee. Constraints (2.12) ensure the compatibility of designating an employee $e\in E$ to work in sector $a\in A$. Finally, constraints (2.13)–(2.15) define the domain of the variables.

2.5 Computational results

The model was coded in Python using PuLP (Mitchell, O'Sullivan, and Dunning, 2011) and executed on an Intel i5-8250U 1.60 GHz with 8 GB of RAM running Windows 10. CBC (`https://github.com/coin-or/Cbc`) from COIN-OR (Lougee-Heimer, 2003) was used as a MILP solver. A time limit of 600 seconds was imposed for each execution of the model.

Two scenarios were considered in our testing. Scenario I assumes that the employees are only eligible to work on the same sectors associated with the solution generated by the company, i.e., the values of c_{ea} are taken directly from such solution. The goal is to verify whether it is possible to obtain an improved solution while keeping the same groups of employees per sector. In Scenario II, we use the values of c_{ea} provided by the company, which were determined according to the ability of a given employee $e\in E$ to work on a specific sector $a\in A$, thus allowing new groups of employees to be formed.

Table 2.4 compares the results produced by the company with those obtained in each of the two scenarios. It can be verified that solution found in Scenario I yielded an improvement of 9% with respect to the solution generated by the company after the pandemic. This can

be considered a very good result, given that the groups of employees are the same in both solutions. Nevertheless, the gains in Scenario II were way more substantial. More precisely, allowing the model to form new groups of employees, while respecting their corresponding skills, led to an improvement of 37.3% on the value of Δ. We also mention that the CPU time in seconds spent by CBC in Scenarios I and II were 0.58 and 0.91, respectively, so the model was very quick in solving the instance.

The average values of R are the same for all scenarios, equaling the one obtained by the solution generated by the company after the pandemic. Note that it is not possible to improve the value of R any further because of constraints (2.10), which in turn impose a certain amount of working hours for each shift $s \in S$ and each sector $a \in A$.

TABLE 2.4: Comparison between the solutions produced by the company and by the model

		Before		**After**		**Scenario I**		**Scenario II**	
a	w	Δ	R	Δ	R	Δ	R	Δ	R
1	1	0	3.70	1	2.00	14	2.00	8	1.83
	2	0		28		14		8	
2	1	0	3.00	10	1.00	30	1.00	10	1.50
	2	0		30		10		20	
3	1	1	5.70	16	2.50	5	2.50	5	2.00
	2	1		9		16		6	
4	1	2	5.00	12	2.00	11	2.00	11	2.00
	2	2		20		11		11	
		Tot = 6	Avg = 4.4	Tot=126	Avg = 1.9	Tot = 111	Avg = 1.9	Tot = 79	Avg = 1.9

Table 2.5 shows the optimized solution found for sector 1, considering scenarios I and II. Note that the solution obtained is used in a cyclic fashion, i.e., one first determines the schedule for 2 weeks and then one repeatedly uses this periodically as long as there are no changes in the personnel. We remark that the company did not face any difficulties during the practical implementation of the optimized solution.

We also conducted experiments on 27 randomly generated instances by varying the number of employees (25, 35, and 45) and sectors (3, 4, and 5), as well as by considering different probabilities for having $c_{ea} = 1$, namely, 65%, 75% and 85%. In this case, the goal is to measure the performance of the model in terms of scalability, solution quality and risk factor when trying to solve larger instances with different characteristics.

Table 2.6 presents the results found on these instances, where for each of them we report the CPU time in seconds, together with the gap, Δ and R values, respectively. One can see that only 18 instances were solved to optimality, but the average gaps were, in most cases, relatively small (below of equal to 5%), with the exception of instances E25-A5-85, E35-A5-65, E35-A5-75, E35-A5-85, E45-A5-75 and E45-A5-85.

We also performed a further analysis on how the value of Δ varies as the density of the skill matrix increases. For this testing, we have generated more instances with other probability values for c_{ea}, considering the case involving 4 sectors. The results of the experiments are presented in Figure 2.2. In the x-axis we report the probability for c_{ea} to assume value 1, while in the y-axis we provide the corresponding Δ values obtained. The graph shows how the value of Δ decreases as the skill matrix becomes more dense. When considering 25 and 35 employees, the reduction is more prominent for up to 75%, whereas the reduction is more consistent for 45 employees.

It is important to highlight that, for practical reasons, we decided to use CBC as a MILP solver, which is open-source and so easily adoptable in many applications. Nonetheless, for the sake of comparison, we also performed tests using Gurobi 9.02 (Gurobi Optimization,

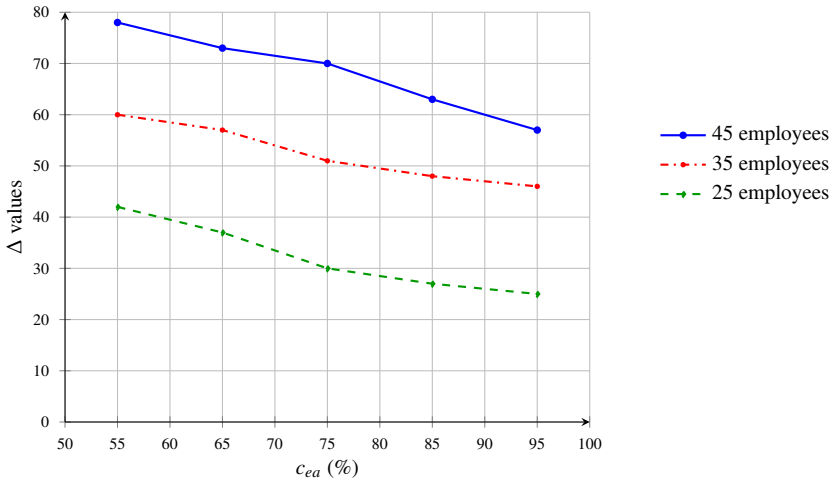
TABLE 2.5: Personnel scheduling for sector 1 in scenario I and scenario II

Personnel scheduling for sector 1 - Scenario I																		
Week 1																		
e	06-07	07-08	08-09	09-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	$s=3$	$\sum h_e$	h_e^{max}	Δ_{e1}
1																40	30	10
2																30	30	0
3															6	36	40	4
4																40	40	0
5																30	30	0
6																30	30	0
Week 2																		
e	06-07	07-08	08-09	09-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	$s=3$	$\sum h_e$	h_e^{max}	Δ_{e2}
1																30	30	0
2																40	30	10
3																40	40	0
4															6	36	40	4
5																30	30	0
6																30	30	0
Personnel scheduling for sector 1 - Scenario II																		
Week 1																		
e	06-07	07-08	08-09	09-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	$s=3$	$\sum h_e$	h_e^{max}	Δ_{e1}
2																30	30	0
3															6	36	40	4
4															6	36	40	4
6																30	30	0
8																40	40	0
16																40	40	0
Week 2																		
e	06-07	07-08	08-09	09-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	$s=3$	$\sum h_e$	h_e^{max}	Δ_{e2}
2																30	30	0
3																40	40	0
4																40	40	0
6																30	30	0
8															6	36	40	4
16															6	36	40	4

TABLE 2.6: Results found by the proposed model on 27 randomly generated instances

Intance	**Time**	**Gap**	Δ	R	**Intance**	**Time**	**Gap**	Δ	R
E25-A3-65	0.21	0.00	63	3.17	E35-A4-85	1.07	0.00	47	3.38
E25-A3-75	0.24	0.00	60	3.17	E35-A5-65	600	0.27	74	2.50
E25-A3-85	0.49	0.00	59	3.17	E35-A5-75	600	0.10	66	2.50
E25-A4-65	0.58	0.00	29	2.13	E35-A5-85	600	0.09	54	2.50
E25-A4-75	0.24	0.00	29	2.13	E45-A3-65	0.36	0.00	106	6.50
E25-A4-85	0.31	0.00	29	2.13	E45-A3-75	0.80	0.00	93	6.50
E25-A5-65	600	0.03	73	1.50	E45-A3-85	0.54	0.00	84	6.50
E25-A5-75	600	0.05	73	1.50	E45-A4-65	0.99	0.00	63	4.63
E25-A5-85	600	0.19	37	1.50	E45-A4-75	1.65	0.00	60	4.63
E35-A3-65	0.33	0.00	48	4.83	E45-A4-85	3.13	0.00	58	4.63
E35-A3-75	0.38	0.00	47	4.83	E45-A5-65	600	0.03	92	3.50
E35-A3-85	1.80	0.00	38	4.83	E45-A5-75	600	0.07	92	3.50
E35-A4-65	0.50	0.00	57	3.38	E45-A5-85	600	0.06	77	3.50
E35-A4-75	1.52	0.00	50	3.38					

FIGURE 2.2: Influence of c_{ea} on the objective function



2020), and it was observed that the optimal solutions for all instances were found in up to two seconds.

2.6 Concluding remarks

This chapter studied a real-life personnel scheduling problem, motivated by Covid-19 pandemic, from a large Italian pharmaceutical distribution warehouse operated by Coopservice. A MILP formulation was proposed for the problem, which was solved using an open-source optimization software, namely CBC from COIN-OR. The optimal solution obtained by our formulation was capable of considerably improving the schedule adopted by the company, which has started to create a plan to exchange operators using the holiday plan starting from June 2020, in order to maintain the distance of at least 15 days between new contacts. Tests were also performed on larger and randomly generated instances to measure the scalability of CBC when solving our model, and the results found were of high quality for the vast majority of the cases. In addition, we conducted experiments using Gurobi 9.02, which quickly managed to find the optimal solutions for all instances. Nevertheless, for larger warehouses with a considerable amount of employees, one may consider using heuristics if the model is not able to scale up.

Because Coopservice spends a large part of its budget on human resources (more than 20,000 employees), it is crucial to have a strategy to properly manage the personnel, especially during emergency situations such as epidemic crises. In general, this problem faces the issue of reducing the risk of contagion, and of organizing the schedule of the shifts in these type of scenarios. Therefore, our findings can be of general interest, especially under the circumstances related to pandemics. The solution of this problem in the specific case of Coopservice is not only relevant for demonstrating a successful case study, but is also interesting for many related applications. For example, the problem of organizing activities in close spaces or in warehouses with aisles containing highly skilled employees that are not allowed to work together.

Other promising research avenues may include the development of a bi-objective model by adding the risk of contagion in the objective function. Moreover, we believe that the model can be extended to produce a fair work-shift distribution. Since in the warehouse there are people with more expertise than others, which is an attribute that goes beyond the skill matrix, one may consider inserting a factor to balance the expertise within a shift in the objective function (Fırat and Hurkens, 2012).

Chapter 3

Scheduling problem for distributed services in hospitals[1]

This chapter addresses a real-life task and personnel scheduling problem arising in Coopservice that needs to provide cleaning services inside a hospital. In this case study, the challenge is to determine a schedule of the employees to clean the whole hospital aiming to minimize the total labor cost, taking into account the fact that the building is a complex structure with multiple levels and each room has different peculiarity. To solve the problem, we propose a three-step approach using mathematical models and metaheuristic algorithms. The solution obtained indicates that the schedule attained by our method is better than the one generated by the company. In addition, to test and validate our approach more thoroughly, a set of artificial instances have been created. The results indicate that our method can help organizations to quickly generate and test a large variety of solutions. Our findings can be of general interest for other personnel scheduling problems involving distributed services.

3.1 Introduction

One of the key operational processes in hospitals is the organization of the cleaning tasks that need to be performed by employees regularly, to maintain the necessary level of cleanliness and sanitary security. Scheduling cleaning tasks is a complex problem (Bartolini, Dell'Amico, and Iori, 2017) that belongs to the general field of personnel scheduling (Bergh et al., 2013); it is a very crucial activity for hospitals, especially in this moment, in which we are facing the Covid-19 pandemic, which calls for an additional level of sanification. Our interest originates from the activity of Coopservice, a large third-party service provider that is responsible for the cleanliness and sanitary security of a large number of hospitals in Italy. However, the problem can be generalized to a large variety of services that should be carried out in a distributed manner across large facilities. To provide their services, third-party companies must take part in tender processes for cleaning services. To achieve success, they have to submit cost-efficient offers. In a hospital, there are often many costs that cannot be influenced substantially by the company. This means that personnel costs and personnel management become the critical factor. Therefore, an efficient and automated planning of personnel schedules enables one to create a competitive advantage in tender processes.

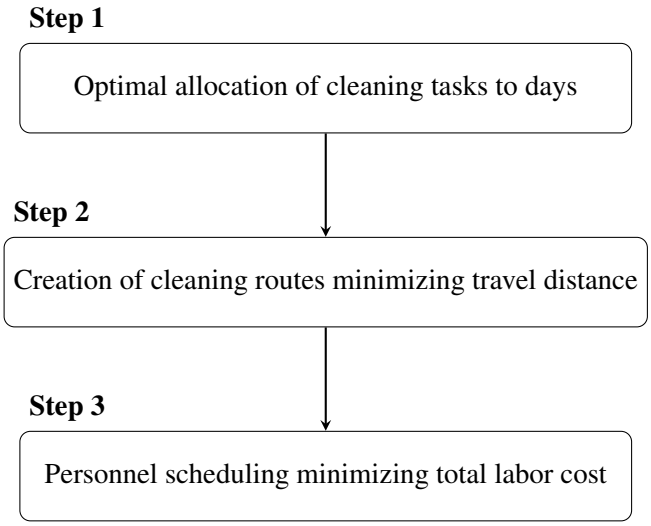
To solve the problem, we need to create cleaning schedules for a specific planning period (e.g., a month), and then allocate the available employees to the schedules by minimizing the total labor cost while satisfying given services requirements. In view of this, we propose a hybrid three-step approach based on the combined use of heuristics and mathematical models.

[1]This work has been published as: Campana, N. P., Zucchi, G., Iori, M., Magni, C. A., & Subramanian, A. (2021). An integrated task and personnel scheduling problem to optimize distributed services in hospitals. In Proceedings of the 23th International Conference on Enterprise Information Systems (ICEIS 2021), 1, 461-470.

As inputs, we are given a set of employees, as well as a set of locations, each with an associated set of cleaning tasks with some physical characteristics, such as the level of sanitary risk, location size and time windows. The first step focuses on a standard one-week period of work. In this period, we first allocate the cleaning tasks to the days so as to balance the workloads. This is achieved by solving a mixed-integer linear programming (MILP) model. Secondly, for each day in the period, we create a set of cleaning routes by solving a vehicle scheduling problem with time windows (VSPTW). The cleaning routes are sequences of visits to locations within the given time windows. Due to its NP-Hardness, we solve the VSPTW by means of an adapted variable neighborhood descent approach, using some well-known local search procedures (Bräysy and Gendreau, 2005a; Bräysy and Gendreau, 2005b). At the third step, we solve a personnel scheduling problem (PSP). Considering a larger one-month period and incorporating further operational constraints to solve the problem, we develop constructive and local search heuristics aimed at assigning the cleaning schedules to the employees and minimizing the total labor cost.

An overview of the three-step approach is presented in Figure 3.1.

FIGURE 3.1: Three-step approach



Our three-step approach was used to solve a real instance derived from a real-world Italian hospital operated by Coopservice, and the results obtained were positively confirmed by the company's managers. Moreover, to test and validate our approach a set of artificial instances have been created. For all instances, we attained full coverage of the cleanliness requirements of the hospital, thus satisfying the needed sanitary level.

The main contribution of this work is to propose a decomposition method to solve a case-study that can help companies to quickly generate and test a large variety of solutions for integrated task and personnel scheduling for distributed services, ensuring the minimization of the costs, in terms of working hours, for the company. In addition, it could enable companies to reorganize cleaning activities, as entire departments and locations can change their risk level because of the COVID-19 emergency. Moreover, it can be used to quickly reorganize the scheduling of employees in case some workers are subjected to quarantine periods due to the infection.

The remainder of this chapter is organized as follows. Section 3.2 gives a formal problem definition. Section 3.3 provides a brief literature review. Section 3.4 describes our three-step approach. Section 3.5 reports the results of our algorithm and, finally, Section 3.6 presents the concluding remarks.

3.2 Problem definition

In a hospital complex, we have a set of buildings B, and every building $b \in B$ also has an internal floor division L^b. For each floor $l \in L^b$, there is a set U_l^b of locations to be cleaned. Each location $v \in U_l^b$ on a floor l of a building b has an associated type (e.g., elevator, office, toilet, etc.) and risk level, which can be classified as low, medium, or high. These two attributes have an impact on both the weekly cleaning frequency and effort. For example, a toilet has to be cleaned more often than a simple office.

Let $D = \{1,\ldots,7\}$ be the set of days. The weekly frequency f_v for cleanness of each location v is provided by the hospital, in other words, f_v gives the number of days in the week in which v has to be cleaned. The company generates a set P_v of feasible patterns, which are the possible combinations for the weekly assignment at location v. When a pattern p is assigned to the location v, all the cleaning tasks of v must be executed in each of the days in the pattern. For example, assuming that $f_v = 3$, we could have the following pattern set: $P_v = \{\{1,3,5\},\{2,4,6\},\{3,5,7\}\}$. In this case, if the company selects the first pattern, then the cleaning tasks at location v are scheduled to be performed on days 1, 3 and 5.

The problem is rather complex; thus, we describe it by using three steps, which also correspond to the main steps of our solution algorithm described later in Section 3.4.

Step 1. For each location v, we define a set of cleaning tasks T_v; each task $k \in T_v$ has a total duration of q_{vk} minutes. In practice, the duration varies according to the type of task, which can be either a complete cleaning or a fast cleaning, and with the size of the location (expressed in m^2) and its type. Moreover, each task $k \in T_v$ also has a time window $[a_{vk}, u_{vk}]$, which determines the earliest and latest times to execute task k. The company wishes to decrease the total amount of cleaning time during weekends (e.g., on days d = 6 and d = 7) by a factor β_d, due to contractual constraints with the employees, and also because the activities of the physicians are usually reduced.

Step 2. For each building $b \in B$, we are given a graph $G^b = (N^b, A^b)$, where the set of nodes N^b is composed of its starting and ending node where all personnel starts and ends the routes, here defined as ρ^b, and the corresponding cleaning locations of b. More precisely, $N^b = \{\rho^b\} \cup \{v : v \in U_l^b, l \in L^b\}$. Each ρ^b of a corresponding block $b \in B$ has an associated time window $[a_{\rho^b}, u_{\rho^b}]$, where a_{ρ^b} and u_{ρ^b} are the earliest and latest departure and arrival times in the day, respectively. The maximum schedule duration was defined, along with the company, as a target duration τ.

Moreover, we assume that all locations on the same floor $l \in L^b$ are interconnected, meaning that there is an arc connecting each pair of nodes located at l. The floors are connected through elevators, which are also cleaning locations. Thus, we define a subset $\hat{N}^b \subset N^b$ composed of all locations that are elevators in the building b. We can then define the arc set as $A^b = \{(i,j) : i,j \in U_l^b, i \neq j, l \in L^b\} \cup \{(i,j) : i,j \in \hat{N}^b, i \neq j\}$. Furthermore, each arc $(i,j) \in A^b$ has an associated traveling distance d_{ij} and time t_{ij}.

Step 3. Regarding the personnel scheduling, the company provides a set of heterogeneous employees E. Each employee $e \in E$ has a contract type C_e, determining the maximum number of days he/she can work in a week (as in our case, where it is set to a month). There are two types of work patterns: (i) "*5+2*", which means five days of work and two days of rest; and (ii) "*6+1*", which consists of six days of work and the Sunday of rest. Note that in the first pattern the days can be arranged in any permutation, whereas in the second one there is only one possible option.

The company also specifies a maximum amount of days C' that any employee can work consecutively without a rest day. This value is useful when the rostering period is longer than a week. In addition, each employee has a maximum amount of working hours per week, defined by M_e. Furthermore, let M' be the maximum working hours in a day, defined by

the Italian law, and $WT^{'}$ be the maximum waiting time between two sequences of cleaning routes. Also, let δ be the number of days associated with the scheduling period.

The objective of the firm is to minimize the total labor cost. Taking the average historic unit cost h (as provided by the hospital) as a proxy for the unit cost of all workers, the objective function is:

$$\min Z = \quad h \cdot z \tag{3.1}$$

where z is the total working time. Let λ_e be the total working time of an employee $e \in E$. Then, minimization of (3.1) boils down to minimization of (3.2) below:

$$\min z = \quad \sum_{e \in E} \lambda_e \tag{3.2}$$

The Integrated Task and Personnel Scheduling Problem (ITPSP) aims at optimize (3.2) while meeting the operational constraints defined at steps 1, 2 and 3.

3.3 Literature Review

The literature on complex OR problems arising in the fields of logistics and scheduling is vast and it is not our goal here to provide a comprehensive review. Instead, we turn our attention to decomposition-based methods closely related to our work.

Considering that we face an integrated task and personnel scheduling problem, we focus our review on approaches that integrate these two aspects. Smet, Ernst, and Berghe, 2016 introduced a problem called Task Scheduling and Personal Rostering Problem (TSRP), which they solved by means of a very Large Neighborhood Search and a column generation algorithm. The TSRP considers non-preemptive tasks, fixed tasks, fixed shifts and qualifications, and aims at minimizing the weighted sum of constraint violations. Indeed, in the TSRP it is often impossible to obey to all the contractual constraints, as they are often imposed by authorities with conflicting priorities. The TSRP differs from the ITPSP that we face for two reasons: (i) the TSRP has five shifts rather than a task demand shift (see Ernst et al., 2004c, for the problem terminology); (ii) in the TSRP only one task can be assigned to a shift. In Lapègue, Bellenguez-Morineau, and Prot, 2013, the authors introduced a very similar problem, the Personnel Task Scheduling Problem (PTSP), and a variant known as Shift Minimization Personnel Task Scheduling problem (SMPTSP) in which the shifts are not fixed and are deduced from the task assignment. The PTSP is a problem in which a set of tasks with fixed start and ending times have to be allocated to a heterogeneous workforce. In Guyon et al., 2010, the authors proposed a decomposition method to solve an integration of the employee timetabling and production scheduling problems. At the first level, they solved a traditional timetabling problem. At the second level, they aimed at supplying feasible schedules for a set of uninterruptible tasks. In Maenhout and Vanhoucke, 2018, the authors proposed a perturbation meta-heuristic for the integrated personnel shift and task re-scheduling problem. In that context, some schedule disruptions can arise as a result of some operational variability, creating the necessity of re-scheduling the already planned roster. More recently, Kletzander and Musliu, 2020 proposed a framework to solve a General Employee Scheduling Problem (GESP), in which a wide range of different constraints needs to be considered to allow the specification of different requirements without the need to introduce a new problem formulation for each variant of the problem. They used an XML format to specify the formulation in a human and machine readable way. The GESP deals with the scheduling of shifts as well as optional tasks and breaks for a set of employees over a certain period of days. Elahipanah, Desaulniers, and Lacasse-Guay, 2013 introduced the Flexible Activity and Task Assignment Problem (FATAP), which takes place in a flexible environment where the detailed activity

and task demands are uncertain, allowing the decision maker to use additional temporary employees, scheduling overtime for regular employees and moving meals break. The authors used a two-phase approach, firstly solving an approximate MILP model and a column generation heuristic embedded into a rolling horizon procedure. In Doi, Nishi, and Voß, 2018, the authors proposed a decomposition-based meta-heuristic algorithm for practical airline crew rostering problems with fair working time. Another interesting paper is the one by Salazar-González, 2014, in which the author developed an arc-flow variable formulation to solve an integrated fleet-assignment, aircraft-routing, crew-pairing problem, and a MILP formulation to solve a crew rostering problem of a Spanish air carrier company.

The ITPSP that we face differs from the problems analyzed in the previous literature because it contains a very general combination of constraints derived from the real-world application at hand. The tasks have an interval time to be executed, but no fixed start time. The shifts are not fixed, and the tasks are non-preemptive. In addition, in the third step, we consider employees with no qualification differences and use the historic average unit cost as a proxy for the unit cost of each worker, so that minimization of total personnel cost is equivalent to minimization of total working time. Lastly, we need to solve this problem for a planning horizon of one month using a standard weekly plan for the tasks to execute.

3.4 Proposed algorithm

Decomposing the problem into multiple steps can be considered a very useful alternative for huge and complex problems (Vance et al., 1997; Juette and Thonemann, 2012; Hoffmann et al., 2017). Therefore, we propose a three-step approach to solve the TPSP, as described in the following.

In the first step, we solve a MILP model that seeks to minimize the maximum number of daily working hours (see Section 3.4.1) to determine a weekly pattern p for each location v. In this phase, one defines all locations that must be cleaned on each day. For convenience, we denote this problem as location scheduling (LS).

Next, for each day, we solve the VSPTW using a local search procedure based on Randomized Variable Neighborhood Descent (RVND), as described in Section 3.4.2. More precisely, the second step aims at minimizing the total travel distance, while respecting the time windows of each location, generating a cleaning schedule for all days.

In the last step, we solve a personnel scheduling problem (PSP), explained in detail in Section 3.4.3. The objective is to organize the available personnel to execute the cleaning schedule generated in the previous step, for a given scheduling period (usually one month), minimizing the total working time of the employees according to (3.2) (and, therefore, minimizing the total cost).

Algorithm 5 provides an overview of the proposed approach.

Our algorithm iteratively executes the three steps mentioned above until a time limit ε is achieved. At each iteration, the method builds a new assignment problem using the information of all previous assignments (line 6). The idea is to generate diversified-yet-efficient weekly patterns, as described in Section 3.4.1. The solution found in the first step is then provided to the second step, where one aims at determining the cleaning schedule by solving a VSPTW (line 7). The third step is responsible for obtaining a personnel schedule using the VSPTW solution as in input (line 8), thus generating a final solution for the TPSP. In case of improvement, the best TPSP solution found is updated (lines 9–11). Finally, the set of weekly patterns is updated before the next iteration (line 12).

Algorithm 5 Iterative three-step algorithm

```
1: procedure 3-STEP(U, E, ε, η, ϕ)
2:     s* ← ∅                                   ▷ Final solution
3:     H ← ∅                                    ▷ Set of weekly patterns
4:     while time limit ε not reached do
5:         m ← buildNewMILPModel(H, U)
6:         WeeklyPattern ← LS(m);
7:         C_Schedule ← VSPTW(WeeklyPattern, η, ϕ)
8:         P_Schedule ← PSP(C_Schedule, E)
9:         if f(P_Schedule) ≤ f(s*) then
10:            s* ← P_Schedule
11:        end if
12:        H ← H ∪ WeeklyPattern
13:    end while
14:    return s*
15: end procedure
```

3.4.1 First step: generating a weekly pattern

In periodic routing problems, choosing a visiting pattern for the customers can be a very useful strategy (Cordeau, Gendreau, and Laporte, 1997). In our case, we are interested in selecting a weekly pattern from P_v for each location v, assigning locations to days. The objective is to minimize the maximum working time across all days.

Let Q_{pd} be equal to 1 if the pattern p contains day d, 0 otherwise. Since some days may have less amounts of work (e.g., at the weekends) we denote as β_d as the balance factor for each day $d \in D$. Let θ_{vd} the time to execute all the tasks of location v in the day d, given by $\theta_{vd} = \beta_d \sum_{k \in T_v} q_{vk}, \forall v \in U, \forall d \in D$. Note that this value does not inform if the day d is present in the patterns of v. Let x_{vp} be the binary variable assuming value 1 if pattern p is assigned to location v, 0 otherwise. In addition, let variable z_1 denote the maximum daily working time. The MILP formulation can be written as follows:

$$\min \quad z_1 \tag{3.3}$$

$$\sum_{p \in P_v} x_{vp} = 1 \qquad \forall v \in U \tag{3.4}$$

$$\sum_{v \in U} \theta_{vd} Q_{pd} x_{vp} \leq z_1 \qquad \forall d \in D \tag{3.5}$$

$$x_{vp} \in \{0,1\} \qquad \forall v \in U, \forall p \in P_v \tag{3.6}$$

$$z_1 \geq 0. \tag{3.7}$$

The objective function (3.3) aims at finding a balanced solution by minimizing the maximum working time across all days. Constraints (3.4) impose that exactly one pattern must be selected for each location v. Constraints (3.5) compute the maximum daily working time. Finally, constraints (3.6) and (3.7) define the domain of the variables.

At each iteration τ of Algorithm 5, the MILP model is modified by inserting a hamming-distance constraint (3.8), which is based on the optimal cuts for two-stage stochastic linear programs with recourse Laporte and Louveaux, 1993. We define ξ as a percentage of how many variables must change with respect to the previous solutions generated. In order to obtain different solutions at every execution of the algorithm, we add the following constraint

to model (3.3)-(3.7)

$$\sum_{(v,p)\in s_\tau} (1 - x_{vp}) + \sum_{(v,p)\notin s_\tau} x_{vp} \geq \xi \qquad s_\tau \in \mathcal{T} \tag{3.8}$$

where s_τ denotes the solution obtained at iteration τ and $\mathcal{T}$ denotes the set of solutions generated during all iterations.

3.4.2 Second step: generating the cleaning schedule

In this step, one has to determine the daily sequence and start time of the visits to locations that the employees have to perform, here called *cleaning schedule*. In our case, this is achieved by solving a VSPTW for each day, where the customers are the locations and the depot is the starting and ending point ρ^b, each of them with an associated time window (see Section 3.2).

To solve the VSPTW, we have designed a heuristic procedure whose pseudocode is described in Algorithm 6. Let N_d be the subset of locations that should be visited on day d, and let $N = \bigcup_{d\in D} N_d$. In addition, let $\mathcal{T}$ be the set containing the VSPTW solutions obtained on each day. Parameters η and ϕ are related to the local procedure, and are described further in this section. For each day, the algorithm generates an initial solution using a greedy approach (line 4), which is possibly improved by a local search procedure (line 5) based on RVND (Subramanian et al., 2010). The solution found after the local search step is then appended to $\mathcal{T}$ (line 6).

Algorithm 6 VSPTW

```
1: procedure VSPTW(N, η, ϕ)
2:     T ← ∅
3:     for d ∈ D do
4:         s ← GreedyAlgorithm(N_d)
5:         s ← RVND(s, η, ϕ)
6:         T ← T ∪ s
7:     end for
8:     return T
9: end procedure
```

The constructive procedure generates an initial solution starting from the lowest to the highest floor, with a view of minimizing the displacement of employees inside the building. At each floor, the task with minimum latest time window is inserted at the last position of the sequence. In case of tie, the algorithm selects the task whose location is the nearest with respect to the last task inserted. If there are no more tasks on the given floor, then one continues the insertion from the next floor. When it is no longer possible to add a task without violating the constraints, a sequence (or a cleaning route) is created and the procedure is repeated until no more tasks are left to be inserted. Note that a daily cleaning schedule is composed of all cleaning routes associated with that day.

The local search engine consists of an adaptation of the RVND procedure. Three classical inter-route neighborhood structures were considered, in particular, cross-exchange, relocate and swap. Whenever an improving move is found, an intra-route local search is applied, also in a RVND-like fashion, using 2-opt and swap neighborhoods. The first improvement strategy is adopted, and the algorithm performs the search in ϕ percent of neighborhood size. The local procedure is iteratively called for up to η times. Furthermore, since a local search move between two floors that are considered far enough is not likely to yield an improving move (e.g., a swap between tasks of the first and ninth floors), we have defined a floor vicinity rule, which only allows moves involving floors $l-1$, l and $l+1$.

Finally, in order to build the cleaning schedule, one should define the optimal start times of the visits by minimizing the total duration of each cleaning route. The timing problem considered here was introduced and solved using the so-called forward slack time procedure by Savelsbergh, 1992. Let $k \in \sigma$ be a task, where σ is a cleaning route composed of n tasks. We are given the following for each task k: (i) q_k is the service time; (ii) ST_k is the earliest feasible start time; (iii) WT_k is the cumulative idle time; and (iv) FD_k is the partial forward slack time. We begin with the following values: $ST_1 = a_1$, $WT_1 = 0$, and $FD_1 = u_1 - a_1$. Next, we compute the remaining values:

$$ST_k = \max(ST_{k-1} + q_{k-1}, a_k) \quad k \in \sigma \tag{3.9}$$

$$WT_k = WT_{k-1} + (ST_k - ST_{k-1} - q_{k-1}) \quad k \in \sigma \tag{3.10}$$

$$FD_k = \min(FD_{k-1}, u_k - ST_k + WT_k) \quad k \in \sigma. \tag{3.11}$$

Finally, the start time of the cleaning route is given by $st^*_\sigma = a_1 + \min(FD_n, WT_n)$.

(The reader is referred to Vidal et al., 2012, for a comprehensive review on timing problems.)

3.4.3 Third step: personnel scheduling

In the personnel scheduling phase, the objective is to minimize the total working time (and, therefore, the total cost) by assigning employees to duties, which is in turn derived from the cleaning routes generated in the second step. Algorithm 7 describes the procedure developed to the referred subproblem.

Algorithm 7 Overall Heuristic

1: **procedure** PSP($\mathcal{T}, E$)
2: $C \leftarrow CreateDailyDuties(\mathcal{T})$
3: $E' \leftarrow AssignEmployeesToDuties(E, C')$
4: $E^* \leftarrow LocalSearch(E')$
5: **return** E^*
6: **end procedure**

Firstly, one determines the duties associated with a day of work, which is composed of a set of one or more cleaning routes (line 2). In particular, the algorithm employs the best fit strategy by maximizing the number of tasks in the duty, always satisfying the constraints of the problem, and avoiding intersections between the times specified in the cleaning schedule. Note that a duty may comprise more than one cleaning route, and the process of putting them together can be seen as a packing problem Krishnamoorthy, Ernst, and Baatar, 2012. The total time of a duty is given by the sum of the execution times of all cleaning routes plus the possible waiting times between them.

Secondly, the algorithm assigns the employees to the duties (line 3). More precisely, the available employees are sorted in non-ascending order according to the their contractual working time. A first-fit policy is then applied for each day, assigning employees to the duties, followed by a local search (line 4). The procedure first tries to interchange the weekly duties between two different employees (as long as they have different M_e). If this change fails to yield a feasible improvement, the algorithm attempts to perform another move involving same pair of employees. In this case, a day is chosen at random, and the respective duties from that day are interchanged between the employees. The procedure terminates when no improving move is obtained.

In Figure 3.2, we present a graphical representation of the proposed algorithm in order to help the reader understand the presented methodology.

3.5 Computational experiments

To validate the proposed algorithm, we solved a real-life instance of the company, and to evaluate the robustness of the method, we created 20 random instances based on real data. The algorithm was coded in Python 3.7.3 and executed on an Intel(R) Xeon(R) CPU E3-1245 v5 3.50GHz with 32GB of memory, running Linux Ubuntu 18.04 LTS 64-bits.

The CBC (`https://github.com/coin-or/Cbc`) solver from COIN-OR was used to solve the MILP model presented in Section 3.4.1 (Lougee-Heimer, 2003). A limit of 10 seconds was imposed for the solver. Regarding the overall procedure, a time limit of 1,800 seconds was imposed. To improve performance, we have parallelized the implementation of Algorithm 6 using 4 cores, benefiting from the fact that the VSPTW can be addressed independently for each day.

3.5.1 Parameters

For all experiments, we used the following parameter values, as defined by the company:

- β: 50%;
- μ: 50 meters per minute;
- τ: 480 minutes;
- WT': 20 minutes;
- C': 6 days;
- δ: 14 days.

Moreover, we adopted $\varepsilon = 1,800$ seconds, and $\phi = 10\%$ after conducting preliminary experiments. In addition, the values of the two remaining parameters, η and ξ, were chosen after comparing the results obtained by different combinations with the one provided by the company on the real-life instance, as described in Section 3.5.3.

3.5.2 Instances

The real-life instance was created using BIM (Building Information Modeling), which is a data model that stores different types of information and consists of parametric objects representing building components. Objects may have non-geometric attributes with functional, semantic or topological information (Volk, Stengel, and Schultmann, 2014). In the hospital associated with the real-life instance (see Figure 3.3 and 3.4), there is a building containing 15 floors with a total of 2,422 locations, not uniformly distributed, and divided into 47 types. The instance has 227 locations referred to as elevators. Each location has a coordinate (x,y) in a normalized Cartesian plane.

In addition, we generated 20 artificial instances by simply combining the floors of the real-life instance. More precisely, we randomly chose a subset of floors, always keeping the original information regarding the existing locations on the respective floor, and built a new instance by sorting such selected floors in an arbitrary fashion.

In vehicle routing and scheduling problems, the distance between two locations is often computed using the Euclidean distance. In our problem, we have a three-dimensional building, so we compute the distance between two locations by considering floors and elevators

FIGURE 3.2: An overview of the proposed algorithm

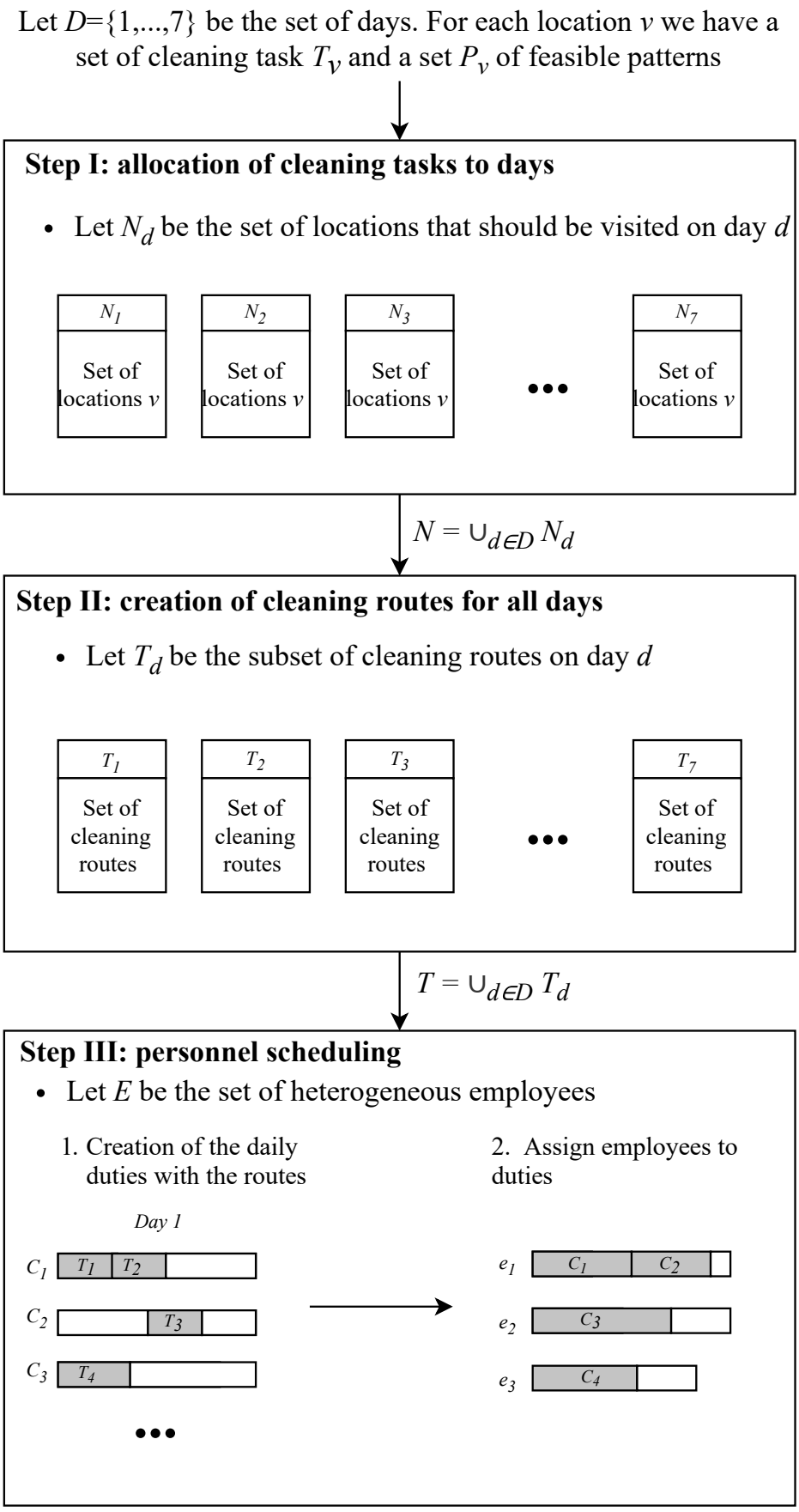


FIGURE 3.3: An overview of the hospital

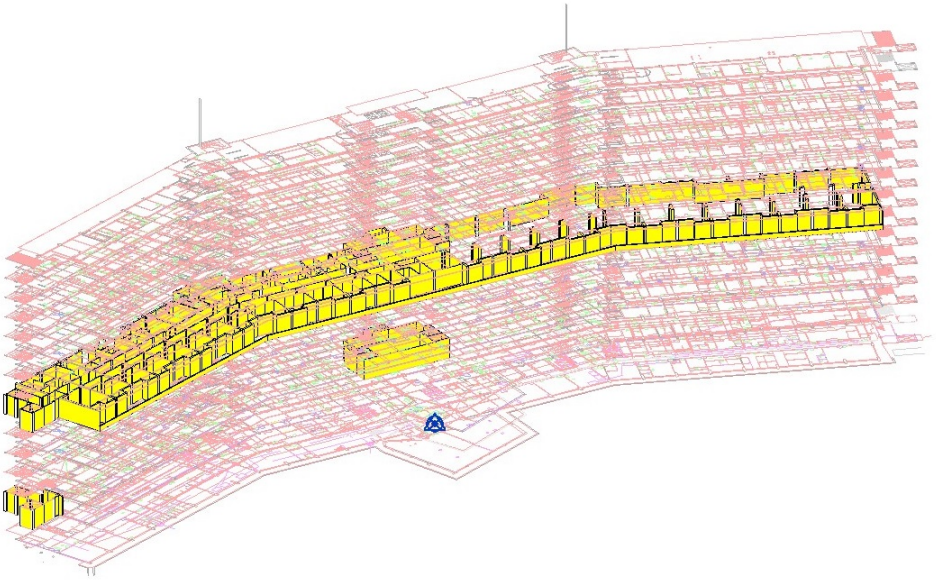
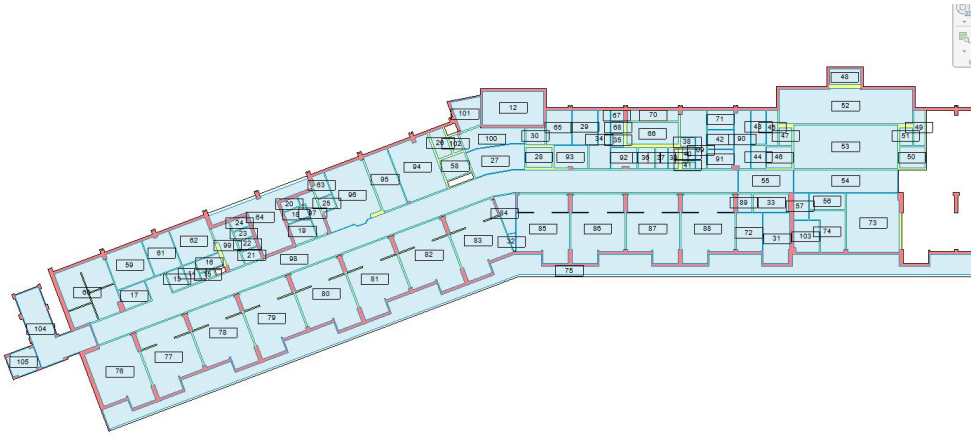


FIGURE 3.4: A detailed view of a department in the hospital



as follows. If the two locations i and j are on the same floor, their distance d_{ij} is computed by the shortest path. Otherwise, we calculate the distance matching the closest elevators on both floors that minimizes the total distance of traveling from i to j. This is done by taking the distance from i to the closest elevator plus the distance from that elevator to j. Let $\mathcal{L}$ be the function that returns the location v of the floor and κ denote an elevator. Function SP in (3.12) calculates the shortest path from i to j:

$$SP(i,j) = \min_{\kappa_1,\kappa_2 \in \hat{N}^b} d_{i\kappa_1}^{\mathcal{L}(i)} + d_{\kappa_2 j}^{\mathcal{L}(j)}. \tag{3.12}$$

The cost c_{ij} is therefore given by:

$$c_{ij} = \begin{cases} SP(i,j) & \text{if} \quad \mathcal{L}(i) \neq \mathcal{L}(j) \\ d_{ij} & \text{otherwise} \end{cases} \tag{3.13}$$

Let μ be the standard walking speed which was empirically observed by the company. The travel time from i to j is given by $t_{ij} = c_{ij}/\mu$.

3.5.3 Results for the real-life instance

The two parameters in which we experimented with different values were η and ξ (η indicates the number of times in which the local procedure is iteratively called and ξ defines a percentage of how many variables must change with respect to the previous solutions generated). We conducted tests for all combinations of the following values: $\eta \in \{20,30,40\}$ and $\xi \in \{1,20\%,50\%\}$. The main objective in testing these combinations is to define the standard parameters for the company to run the algorithm. Table 3.1 shows the results achieved on the real-life instance. For each tested combination of η and ξ, we report the best solution found z computed as in (3.2) and the percentage gain over the solution provided by the company, whose value is $z^c = 135780$. It can be observed that the setting $\eta = 40$ and $\xi = 1$ yields the best improvement of 5.83%. The percentage gain is computed as $100(z^c - z)/z^c$.

TABLE 3.1: Results obtained for the real-life instance

η	ξ	z	%gain
20	1	128,323	5.81
20	20	128,330	5.81
20	50	128,324	5.81
30	1	128,309	5.82
30	20	128,328	5.81
30	50	128,327	5.81
40	**1**	**128,302**	**5.83**
40	20	128,320	5.81
40	50	128,335	5.80

3.5.4 Results for the artificial instances

We also carried out a similar experiment for the artificial instances. For each instance, we considered the same values of η and ξ specified in the previous section.

Each instance created has been run for all the possible combination and by observing the results obtained, and computing the frequency of the parameter setting for which the best solution was found, we found that the best configuration of η and ξ are 20 and 50, respectively.

Table 3.2 reports the information of each instances, namely, the number of floors and locations. In the last two columns, we provide the value of the objective function obtained with the best configuration and the CPU time required to find the best solution. The results show that, on average, 920 seconds are necessary for the method to obtain the best solution.

TABLE 3.2: Best results found on the artificial instances with the best configuration of η and ξ

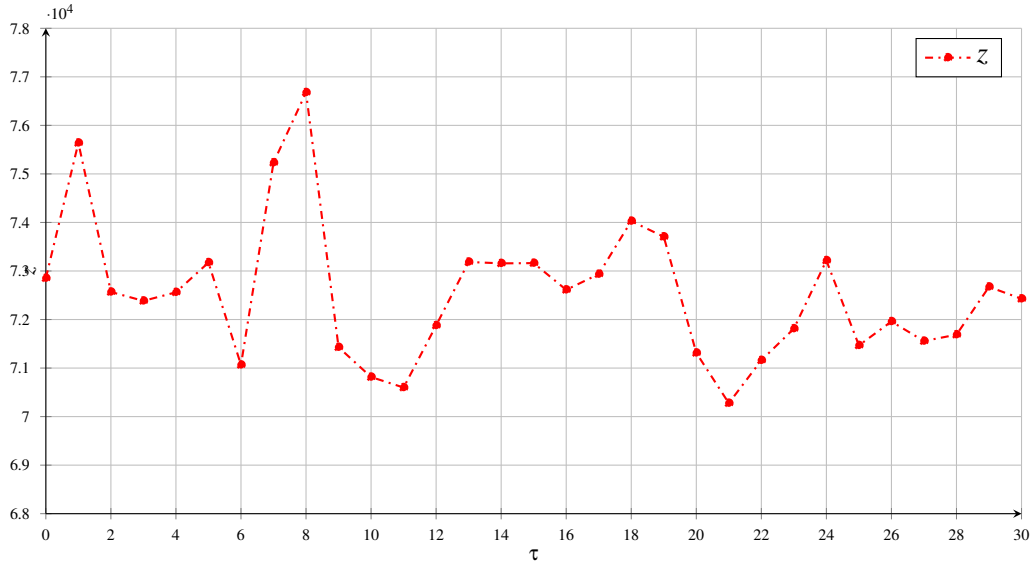
Instance	**Floor**	**Locations**	z	t^*
1	4	534	46326	266
2	11	1678	126496	667
3	6	857	66012	1095
4	3	483	38706	309
5	14	2020	164338	152
6	5	743	50978	979
7	10	1451	106266	1726
8	3	494	40432	741
9	3	439	31564	53
10	11	1613	132760	1620
11	11	1643	128074	730
12	3	399	43854	1776
13	9	1226	105810	86
14	5	777	81802	865
15	15	2191	180734	1624
16	2	223	23732	1800
17	11	1645	126346	261
18	6	897	70277	346
19	13	1915	165054	1509
20	13	1875	150448	1800

Figure 3.5 depicts an example of the behavior of the proposed algorithm on instance 18, when considering the best configuration of parameters found, i.e., $\eta^{best} = 20$ and $\xi^{best} = 50$. The plot shows the importance and effectiveness of using an iterative approach, as illustrated by the results obtained in the last step of each iteration. In this case, it can be observed that the best result is achieved at iteration 21.

3.6 Conclusions

In this work, we proposed an iterative three-step procedure to solve a real-life problem of integrated task and personnel scheduling aiming to minimize the total labor cost. By implementing this approach, the solution obtained by our formulation was capable of improving the schedule adopted by the company in an acceptable CPU time for this very complex problem. Tests were also performed on larger and randomly generated instances to measure the scalability of the proposed method and to find a combination of η and ξ that can produce the best results based on the dimension of the building. The results are preliminary and yet they are very positive from the company perspective, which is now able to generate and simulate many different scenarios. Because Coopservice spends a large part of its revenues on human resources (more than 20,000 employees), it is crucial to have a strategy to properly manage the personnel, especially during emergency situations such as epidemic crises (Zucchi, Iori, and Subramanian, 2021). Future research might include: (i) the insertion of more complex cost functions in (3.1) that could take account of different employees' levels; (ii)

FIGURE 3.5: Variation of z for each iteration τ in the instance number 18



a sensitivity analysis for the many parameters used in models and algorithms; (iii) a deeper evaluation of the importance of each key parameter making use of the Finite Change Sensitivity Index (FCSI) (Borgonovo, 2010a; Borgonovo, 2010b); (iv) a more profound economic cost analysis related to the labor cost assuming it differs across workers and changes over time, meaning that one should take into account a possible evolution of h, and consider the total profit (or the total value) as the objective function for a given (sufficiently large) span of time. Moreover, to improve performance in terms of solution quality, one could incorporate metaheuristic strategies, such as perturbation schemes in Algorithms 6 and 7, with a view of obtaining better local optimal solutions.

Chapter 4

Lead time forecasting for a pharmaceutical supply chain[1]

Purchasing lead time is the time elapsed between the moment in which an order for a good is sent to a supplier and the moment in which the order is delivered to the company that requested it. Forecasting of purchasing lead time is an essential task in the planning, management and control of industrial processes. It is of particular importance in the context of pharmaceutical supply chain, where avoiding long waiting times is essential to provide efficient healthcare services. The forecasting of lead times is, however, a very difficult task, due to the complexity of the production processes and the significant heterogeneity in the data. In this paper, we use machine learning regression algorithms to forecast purchasing lead times in a pharmaceutical supply chain, using a real-world industrial database. We compare five algorithms, namely k-nearest neighbors, support vector machines, random forests, linear regression and multilayer perceptrons. The support vector machines approach obtained the best performance overall, with an average error lower than two days. The dataset used in our experiments is made publicly available for future research.

4.1 Introduction

Long waiting times for service interventions are a recurring feature in the health sector, especially for public services. Clearly, timely treatments and drug administrations are crucial factors for improving the quality of healthcare services, and often also for saving the lives of patients, mainly in emergencies (Brown et al., 2016; Tetteh, 2019). The delay for medical interventions, whether through medication, diagnosis or surgical procedures, can indeed aggravate pathologies, given the possibility of deterioration of health conditions over time. Longer waiting times for medical intervention can increase readmission rates as well (Moscelli, Siciliani, and Tonei, 2016). Nowadays, this is even more crucial because of the recent COVID-19 pandemic, which is causing an increase in the number of pharmaceutical products urgently required by the many patients affected by the disease ("Coronavirus disease 2019 (COVID-19): A literature review" 2020).

Among other factors, long waiting times for receiving medicines can be associated with delay in the administrative packaging, logistic problems with tracking and delivery (Haugh, 2014) and several other factors that could be outside the control of patients or healthcare professionals. Within this scenario, the analysis and proposition of measures to reduce waiting times for all possible related factors is important in healthcare policy guidelines (Moscelli, Siciliani, and Tonei, 2016). The availability of medicines in healthcare service networks,

[1]This work has been published as: de Oliveira, M. B., Zucchi, G., Lippi, M., Cordeiro, D. F., da Silva, N. R., & Iori, M. (2021). Lead Time Forecasting with Machine Learning Techniques for a Pharmaceutical Supply Chain. In International Conference on Enterprise Information Systems (ICEIS 2021), 1, 634-641.

pharmacies and hospitals is directly related to the lead time of the supply chain (Tetteh, 2019).

Our work is motivated by the activity of a logistic company, Coopservice group, that receives the pharmaceutical products from the suppliers and then organizes the shipping, when needed, to the healthcare facilities. To organize the service in the best possible way, it is crucial for the company to correctly estimate the purchasing lead time, that is, the time that is elapsed between the moment in which an order for a good is sent to a supplier and the moment in which the good is delivered to the company. Correctly forecasting this purchasing lead time (lead time for short, in the following) in the supply chain of the pharmaceutical sector is a crucial task, as it largely affects the whole industrial process of the healthcare services. In addition, proper estimation of lead time is a critical parameter in the relationship between the management process and the customer (Noori-Daryan, Taleizadeh, and Jolai, 2019), being lead time one of the most important performance indicators for the management of manufacturing and service production processes (Kim, Kim, and Lee, 2014). Furthermore, accurate forecasting of lead times can assist in optimizing the production processes, by more accurately selecting the needed quantities and thus shortening the overall production times (Gyulai et al., 2018).

Besides, lead time prediction is a crucial aspect to keep under control in the pharmaceutical supply chain, because sometimes having the medicine available at the right time can save lives. Lead time forecasting could allow the pharmaceutical companies to predict and to avoid possible out of stock, caused by a supplier. Besides, based on the lead time, it is possible to evaluate the different suppliers and select the best ones. In addition, with a good prediction it is possible for the pharmaceutical companies to define different level of security stock of the goods for each month, making the procurement process leaner and more cost effective.

However, lead time forecasting is an extremely challenging task. In general, the estimation of lead times from historical data has been a recurrent issue in the literature since the 1960s, and even in recent years some traditional systems simply obtain lead time by computing average values based on historical data, with the result of deficiencies in production planning and control (Lingitz et al., 2018). The proposed approaches in this research field can be divided into conventional methods and intelligent methods, with the former not using artificial intelligence and the latter exploiting data mining and machine learning. In both cases, data used for experimental evaluation can be real and/or simulated. In this research, we exploit intelligent methods, leaving conventional methods to an analysis of the literature.

Recently, there have been significant advances in this research field using artificial intelligence (Ioannou and Dimitriou, 2012; Gyulai et al., 2018). This process is mainly due to the growing availability of large data collections in different fields of manufacturing, that can enable data-driven technologies such as machine learning, data mining, knowledge discovery in databases, and big data analytical tools Fayyad, Piatetsky-Shapiro, and Smyth, 1996; Tsai, Lin, and Ke, 2016; Frank, Dalenogare, and Ayala, 2019; Kabugo et al., 2020). Nevertheless, most of the intelligent techniques used in recent research do not make use of real data (Öztürk, Kayalıgil, and Özdemirel, 2006), while using computer simulations to generate data and considering many simplifying assumptions for the internal manufacturing process.

Given the limitations of the methods mentioned so far, in this paper we aim to use intelligent methods to predict the delivery times of suppliers who have to deliver the goods to a company that manages the pharmaceutical supply-chain of hospitals. To this aim, we compared five different machine learning regression approaches, namely: k-nearest neighbors (KNN), support vector machines (SVM), random forests (RF), linear regression (LR) and multilayer perceptrons (MLP).

The use of accurate lead time forecast can be highly beneficial in the planning of both production and logistic services in the pharmaceutical field. We mention, to this regard,

the work by Gatica, Shah, and Papageorgiou, 2001, who studied stochastic aspects related to product development and capacity planning in the pharmaceutical sector, by proposing a multistage stochastic programming approach, and that of Kramer, Cordeau, and Iori, 2019, who proposed a metaheuristic algorithm for the delivery of pharmaceutical products in the region of Tuscany (operated by the Coopservice group). In the former work, accurate prediction of the lead time for purchasing the products could be used within the what-if analysis, while in the latter work, accurate predictions could be used to define the starting points of the deliveries, as multiple depots are available, and the possible use of temporary depots at the hospitals, so as to reduce transportation costs and times.

The reminder of the paper is organized as follows. In Section 4.2 we present the related works and compare our work with the literature. In Section 4.3 we briefly present the classic techniques that we used to predict the lead time. In Section 4.4 we describe the dataset used in the experiments, which are illustrated in Section 5.6. Finally, Section 5.7 concludes.

4.2 Related works

In an Industry 4.0 scenario, big data analytics can be divided into five different categories: predictive, descriptive, inquisitive, preventive and prescriptive analytics. Predictive analytics aims to anticipate what will happen in the future: descriptive analytics instead provides information and explanations about what has happened; inquisitive analytics tries to answer why it has happened, and preventive analytics provides insight to understand what is necessary to be done. Finally, prescriptive analytics provides information for decision-making (Sivarajah et al., 2017; Cabrera-Sánchez and Villarejo-Ramos, 2020). Big data analytics is very often associated with artificial intelligence, data mining, and machine learning instruments (Dean, 2014), with the aim to develop systems that can automatically extract information and discovery patterns in large data collections (Lu et al., 2015; Kuo, Lin, and Lee, 2018), so as to provide beneficial insights to decision makers (Chamikara et al., 2020).

By mid-1980s, many studies on operating and lead time estimation through mathematical formulations, as well as statistical methods with analysis of variance (ANOVA) were proposed (Chang, 1997; Tatsiopoulos and Kingsman, 1983). Forecasting through mathematical modeling approaches has also been recently proposed for a custom system disregarding the current system workload (Vandaele, Boeck, and Callewier, 2002). In a more complex product development scenario, a heuristic approach was proposed, by explicitly modeling networks of operating system activities (Jun, Park, and Suh, 2006). Other research has proposed the use of queuing networks for lead time analysis and prediction (Ioannou and Dimitriou, 2012; Berling and Farvid, 2014) with the use of discrete event simulation through mathematical expressions, assuming a continuous demand and studying the variance of the lead time. Conversely, a case-based reasoning approach was proposed in Mourtzis et al., 2014 to predict the lead time of complex engineered-to-order products. Pfeiffer et al., 2016 made use of multivariate regression statistical methods using simulated data to obtain the production lead time of a flow-shop system.

Mathematical and statistical formulas were reformulated and proposed for production lead time estimates in chemical sector modular production plants (Sievers et al., 2017). However, the main disadvantage of all the methods cited so far is that they consider that past trends could possibly be repeated in the future (Öztürk, Kayalıgil, and Özdemirel, 2006; Ioannou and Dimitriou, 2012). Moreover, there are few researches evaluating the interactions of supply chain elements such as lead times and forecasting procedures (Sievers et al., 2017; Hosoda and Disney, 2018; Lingitz et al., 2018; Goltsos et al., 2019). Additionally, databases generated by simulation often consider a perfect production system, without introducing machine breakdowns, maintenance downtime and raw material delays (Lingitz et al., 2018).

When performing lead time analysis and forecasting, it is important to consider external factors too, such as relationships and interactions between different supply chains (Hosoda and Disney, 2018; Ponte et al., 2018; Goltsos et al., 2019; Noori-Daryan, Taleizadeh, and Jolai, 2019). Chung, Talluri, and Kovács, 2018 showed that lead time prediction is a key factor because the lead time uncertainties can affect service level and order lead time performance. Understanding these dynamics allows companies to reduce their exposure to different types of delivery risk and to better manage their supply chain.

Despite the large amount of works in this area, we could not find comprehensive studies on machine learning algorithms for lead time forecasting in the field of pharmaceutical distributions. Related works are limited to the use of Monte Carlo simulation to predict the production lead time (Eberle, Sugiyama, and Schmidt, 2014), and to the proposal of cyclic production plans combined with outsourcing in the packaging of medicines in the Netherlands (Strijbosch, Heuts, and Luijten, 2002). With this paper we aim at filling this research gap.

4.3 Methodology

As already stated in Section 5.1, we employ a machine learning approach for purchasing lead time forecasting of pharmaceutical services. We formulate the task as a regression problem, where the aim is to predict a single real number $y \in \mathbb{R}$ as a function of a set of features $x \in \mathbb{R}^d$. Supervised machine learning approaches are able to learn a function f that computes a value $\hat{y}$ from a given input vector $\hat{x}$. Such a function is learned from a dataset $\mathcal{D}$, which consists of a collection of N pairs (x_i, y_i) where each input example x_i is associated with the corresponding target y_i, that is the target of the forecasting system. In this work, we compare several simple, classic regression algorithms, largely used in statistics and machine learning applications, with the aim of finding the one that performs the best on our real-world data set, without resorting to more sophisticated approaches. We compare two efficient linear methods, namely linear regression and linear support vector machines, against three simple non-linear ones, namely random forests, k-nearest neighbors, and multi-layer perceptron. We leave the use of more advanced machine learning approaches for future research.

4.3.1 Linear regression

Linear regression (LR) is a widely employed parametric regression technique (Montgomery, Peck, and Vining, 2012), where function f is computed as a linear combination of input features: $f(x) = \beta^T x + \beta_0$. The vector of parameters β is typically learned by minimizing the sum of squared errors on the training set. Clearly, this approach achieves good results when a linear function results to be a reasonable approximation of the dependency relation holding between input and output variables, while suffering when such dependency is strongly non-linear.

4.3.2 Linear support vector machines

Support vector machines (SVM) are a classic machine learning approach that can be used both for classification and for regression. In the regression setting, the goal is to find a function f for which the forecasting error with respect to target y is at most equal to a predefined tolerance threshold ε for the elements in the training set (Drucker et al., 1997). In its linear formulation, which is the one we employ in this paper, the function to be learned is still a linear combination of the features. The optimal (or close to optimal) parameters are found by heuristically solving a constrained quadratic optimization problem (Albers, Critchley, and Gower, 2011).

4.3.3 Random forests

A random forest (RF) is an ensemble classifier that consists in a collection of n different decision trees (Breiman et al., 1984). A decision tree is an interpretable classifier that inductively learns classification rules by testing the informativeness of the attributes (features) with respect to the category (in case of classification) or the target value (in case of regression) to be predicted. Several different decision trees can be obtained either considering different sets of features, or by subsampling different sets of training examples. In the regression setting, the output prediction of the RF is computed as the average of the predictions of individual trees.

4.3.4 k-nearest neighbors

Based on the concept of distance (or similarity) between examples, k-nearest neighbors (KNN) is not properly a learning algorithm. Given a test example x, the KNN algorithm looks for the k examples in the training set that are the most similar to x, i.e., the nearest ones according to a given metric, such as the Euclidean distance, which we use in our experimental evaluation. Once the k nearest neighbors are found, the algorithm computes the prediction as an average, or voting procedure, among them. In a regression setting, the predicted target value $\hat{y}$ is simply computed as the weighted average of the targets y_j of all k neighbors.

4.3.5 Multi-layer perceptron

A multi-layer perceptron (MLP) is a very simple artificial neural network that can learn non-linear functions between input and output variables (Rumelhart and McClelland, 1987). An MLP consists in a stack of layers, each consisting of a certain number m of neurons. The first layer consists of input variables. In the second layer, named hidden layer, the output of each neuron is computed as a non-linear combination of input variables, whose weights are learned during a training phase. Finally, the last layer computes the output of the network as a non-linear combination of the output of the hidden neurons, again with adjustable, learnable weights.

4.4 Dataset

A crucial ingredient of any machine learning application is the preparation of the dataset used for training and evaluation (Ristoski and Paulheim, 2016). The database used in this research was made available by an integrated service company, the Coopservice Group. Founded in 1992, the Coopservice Group provides specialised services to private companies and public entities. The Group operates worldwide, with its headquarters in Italy, and counts around 20,000 employees. It offers a variety of facility services, especially the ones that are not part of the core businesses of the clients, including: industrial, commercial and healthcare cleaning; management and maintenance of buildings and systems; management of energy supplies; security and surveillance; transport and handling of goods; industrial and commercial moving; collection and transport of special waste. With 18 logistic warehouses and a storage area of over 150,000 squared meters, Coopservice Group is the leader in healthcare and pharmaceutical logistics in Italy, and a key provider of management and distribution services for pharmaceuticals, medical-surgical devices and non-medical consumables. The key aspects for the services are relying on a large workforce, working at client-sites, maintaining consistent quality and monitoring performance.

Forecasting lead times is a crucial task for Coopservice, because with an accurate prediction it is possible to optimize and manage the scheduling of the truck deliveries, as well

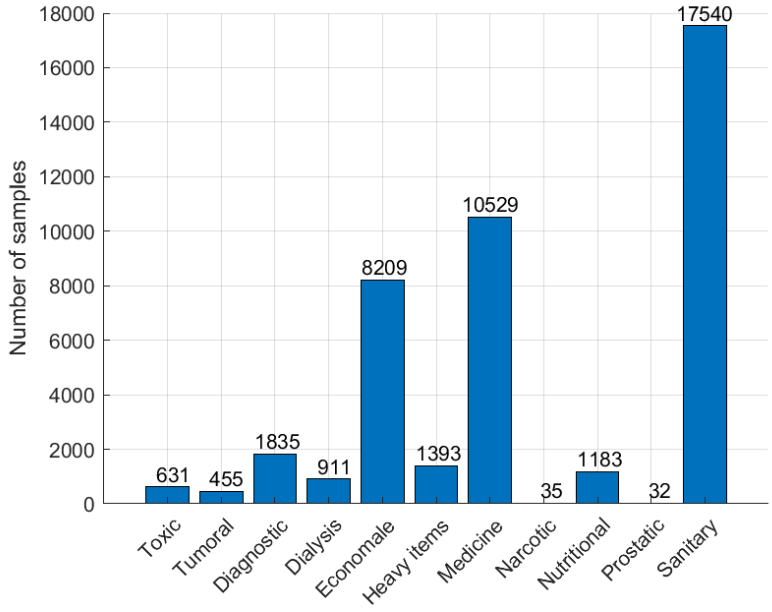


FIGURE 4.1: Distribution of the number of samples in the dataset, for each different category.

as predict the unloading process schedule for the inbound area. Thanks to this, it is possible to better organize the shifts of the employees in the warehouse. In addition, lead time prediction allows the company to have a better knowledge of the supplier and to evaluate its performance. In order to do this, a supplier rating system can be created, considering the historical data and the prediction. Finally, with an accurate forecasting of lead times, the management of safety stock in the warehouse can be safer, avoiding negative events like overstock and stockout.

In the pharmaceutical database provided by Coopservice, the total number of samples was 42,753 collected during 2018.

All pharmaceutical products in the database are associated with some specific categories, namely: tumoral, diagnostic, medicine, nutritional, prostatic, sanitary, dialysis, heavy items, toxic, narcotic, and economale (that are all the non-medical items like pens, papers...). All these categories were used in our study, although most of the data belong to economale, medicine, or sanitary categories, as shown in Figure 4.1.

For each sample in the database, eight independent variables were considered as the input vector x for our machine learning systems used to forecast lead times:

- day of the month of the customer order (1 to 31);
- weekday of the customer order (1 to 5, from Monday to Friday);
- month (1 to 12) of the customer order;
- supplier code identifier;
- product name identifier;
- pharmaceutical product type category;
- ordered quantity (pills);
- distance between supplier and the pharmacy warehouse (km).

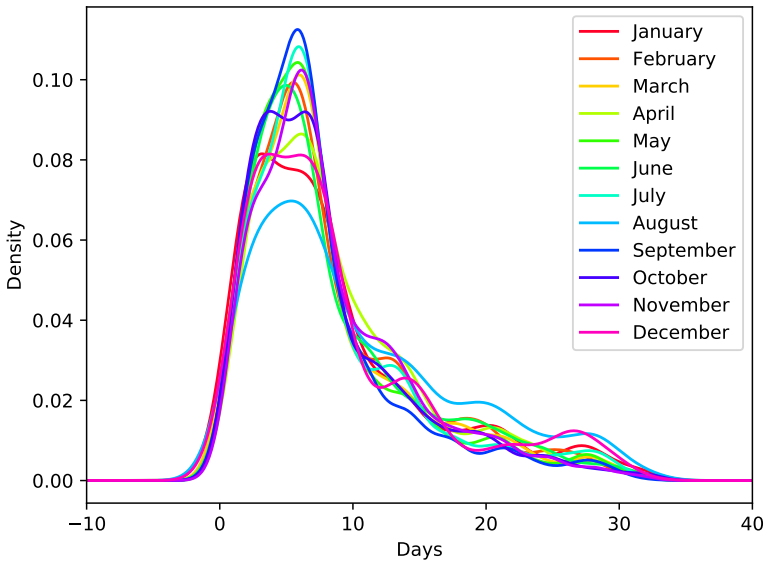


FIGURE 4.2: Lead time distribution, as a function of the month.

A standard pre-processing phase was applied to the database, including explorative data visualization, cleaning and removal of duplicates and corrupted data, outlier detection, manipulation of missing values (Ristoski and Paulheim, 2016). In particular, we used boxplots to identify outliers and extreme values (Hu, Pedrycz, and Wang, 2018; Sagaert et al., 2019) to remove corrupted data. Figure 4.2 shows the distribution of the lead time for each month. It can be noticed that the trend is quite similar for all the months, with a peak between 3 and 7 days, and very few values exceeding 32 days. After a detailed analysis of the cases with such a large lead time, we noticed they were due to insertion errors in the original database, and hence we discarded them. Overall, around 5% of data were removed following the whole pre-processing and cleaning procedure. The resulting dataset is available for research at `https://github.com/regor-unimore/Pharmaceutical-Lead-Time-Forecasting.git`.

4.5 Experimental Results

To compare the machine learning systems employed in our analysis, we performed two different experiments, splitting the whole corpus by category, as well as by month.

Initially, in order to select the best hyper-parameters of each algorithm, we employed a standard 10-fold cross-validation procedure, where the whole dataset is partitioned into 10 different groups, named folds. In turn, each fold is considered as test set, whereas the remaining folds were split into 2/3 for the training set, and 1/3 for the validation set. The training set is the set of examples used during the learning phase to find the optimal model parameters, whereas the validation set is the set of examples that is employed to evaluate the performance of the learned model. In this way, we selected the following hyper-parameters for our machine learning systems: 100 estimators (i.e., number of trees) for the RF, linear kernel and a regularization term $C = 1$ for SVM, a value of k=13 for the number of neighbors in KNN, and a single hidden layer with 3 neurons for MLP.

Then, we performed two distinct experiments. As a first experiment, we partitioned the dataset by category, and we split each portion into 2/3 to be used for training, and 1/3 to be used for test. As a second experiment, we partitioned the dataset by month, and again we

	KNN	LR	RF	MLP	SVM
Tumors	3.37	2.23	2.39	3.87	**1.94**
Diagnosis	4.98	**2.37**	3.40	7.41	2.51
General	4.59	**2.22**	3.48	8.12	2.30
Medicine	4.10	2.22	2.71	5.51	**2.02**
Nutritional	2.90	**2.21**	2.28	4.60	2.28
Prostatic	3.11	**1.75**	3.07	3.15	3.38
Sanitary	3.11	**2.22**	2.49	6.98	2.30
Dialysis	3.23	**1.50**	2.49	2.34	1.83
Heavy Goods	2.66	**1.79**	2.40	5.40	1.86
Toxic	3.73	**1.70**	2.68	2.03	1.73
Narcotics	**3.72**	5.16	5.44	4.29	4.81
Average	3.59	**2.31**	2.99	4.88	2.45

TABLE 4.1: Mean squared error obtained per each different category (best results in bold).

split the data of each month into 2/3 for training, and 1/3 for test. In both experiments, as a standard performance metric, we considered the mean squared error (MSE) as the average of the squared difference between true and predicted lead time: $\text{MSE} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ where y_i is the true lead time, and $\hat{y}_i$ is the forecast value.

The two experiments have different goals. In the first case, one full year of data for each category is used both for training and for test, thus we can evaluate the performance of a forecasting approach when a long period of data is available, for each single category. Conversely, in the second experiment, we take into account all the categories, by partitioning the data by month: in this way, we can evaluate whether data from different categories can help in forecasting the lead times of each product.

As for the first experiment, in Table 4.1 we report the performance achieved by all the competitors on each distinct category. The results show that LR is the best performing method. A very similar performance is also obtained by the SVM approach, that achieves the lowest error in two categories (Tumors and Medicine). Narcotics results to be the most difficult category to forecast, which is not surprising, as it contains very few examples. For that category, KNN is the best-performing algorithm.

In our second setting, the samples of all the categories are used within the training and test set of each month. As shown in Table 4.2, in this case SVM is clearly the best performing algorithm, achieving the lowest MSE in every month, with an average error equal to 1.89 days, which is largely better than the second best approach, which is RF, that achieves an MSE equal to 3.07 days only. Overall, the results of both settings suggest that the use of non-linear approaches does not significantly lower the forecasting error.

4.6 Conclusions

This paper presented a methodology for lead time forecasting in the pharmaceutical supply chain with machine learning techniques. In particular, we compared support vector machines, random forests, multi-layer perceptron, linear regression, and k-nearest neighbors on a very large collection of examples provided by a large company with headquarters in Italy. Our experimental results are very encouraging, showing how the purchasing lead time can be forecast with high accuracy, especially for linear support vector regression. In particular, the use of simple non-linear approaches does not seem to yield significant improvements in the forecasting.

The research described in this paper aims to fill a gap in the scientific literature regarding lead time forecasting for the purchase of pharmaceutical products. An accurate forecast of

	KNN	LR	RF	MLP	SVM
January	3.43	5.13	2.62	5.60	**1.86**
February	2.77	4.20	2.05	5.46	**1.58**
March	3.88	2.83	6.14	6.94	**1.80**
April	3.96	9.51	2.94	8.03	**1.87**
May	3.57	5.74	2.54	7.69	**1.55**
June	3.79	5.91	2.71	7.01	**1.58**
July	3.84	2.69	3.00	8.75	**2.09**
August	4.01	2.43	3.15	13.47	**2.02**
September	3.49	5.47	2.55	6.44	**1.55**
October	3.87	2.36	2.95	7.36	**1.76**
November	3.91	2.72	2.95	6.95	**2.21**
December	4.09	7.01	3.25	10.33	**2.86**
Average	3.72	4.67	3.07	7.84	**1.89**

TABLE 4.2: Mean squared error obtained per each different month (best results in bold).

such lead time can be crucial for decision making, optimization, and planning in the overall pharmaceutical supply chain. Waiting times for drugs and medicines could in fact be reduced, and hospitals and pharmacies could choose the most convenient supplier at every moment on the basis of accurate predictions. This can be very relevant when treating patients with urgent needs, as well is fast-changing medical conditions, as the ones we are currently facing in the COVID-19 pandemic.

Future research will incorporate forecasting of internal supply chain lead times of real service processes. In this way, the forecast of lead time for purchasing products will be coupled with the forecast of the entire supply chain lead time, providing decision makers with a larger instrument of analysis. In addition, more sophisticated approaches to lead time forecasting could be exploited, with simulation of nonlinear systems to investigate how machine faults and maintenance procedures can influence lead time.

Chapter 5

A Metaheuristic Algorithm for a Multi-period Orienteering Problem[1]

This chapter addresses a real-world multi-period orienteering problem arising in a large Italian company that needs to patrol an area in order to provide security services to a set of customers. Each customer requires different services on a weekly basis. Some services are mandatory, while others are optional. It might be impossible to perform all optional services, and each of them is assigned a score when performed. The challenge is to determine a set of routes, one per day, that maximizes a weighted sum of the total collected score and total working time, while meeting several operational constraints, including hard time windows, maximum riding time, minimum number of services performed, and minimum time between two consecutive visits for the same service at the same customer. To solve the problem, we propose an iterated local search that invokes at each iteration an inner variable neighborhood descent procedure. Computational tests performed on a large number of real-world instances prove that the developed algorithm is very efficient, and finds in a short time solutions that are consistently better than those produced by a mathematical model, and those in use at the company.

5.1 Introduction

Every day, private security guards need to inspect structures, parks, buildings, and many other facilities, in order to counter potential criminal actions or simply restore normal safe conditions after breakdowns. In this paper, we study a real-world security problem in which patrols are required to perform a set of services at customers located in a vast area. Some services are mandatory, while others are optional. The optional services, when performed, induce a score. The goal is to maximize the total collected score and minimize the total working time, while meeting a number of operational constraints.

The problem originates from the everyday activity of Coopservice, a large service provider company located in Italy (https://www.coopservice.it/). Counting on more than 25 000 employees, Coopservice operates a number of different services, including logistics, transportation, cleaning, maintenance and security. The company also operates all around Italy car patrolling services for customers who booked their service. Figure 5.1 shows the current customers of the Emilia Romagna region, divided by province.

The customers are geographically dispersed in the area and are consequently divided into clusters. Each cluster is assigned to a patrol, which performs every day a route to visit customers and execute the required services. Figure 5.2 provides better details for the province

[1]Preliminary results of this work appears in: Zucchi, G., Correa, V., Iori, M., dos Santos, A., Yagiura, M. (2022). A Metaheuristic Algorithm for a Multi-period Orienteering Problem Arising in a Car Patrolling Application. In International Network Optimization Conference (INOC 2022), 99–104.
The full version of the paper is now submitted to Networks, an internation journal.

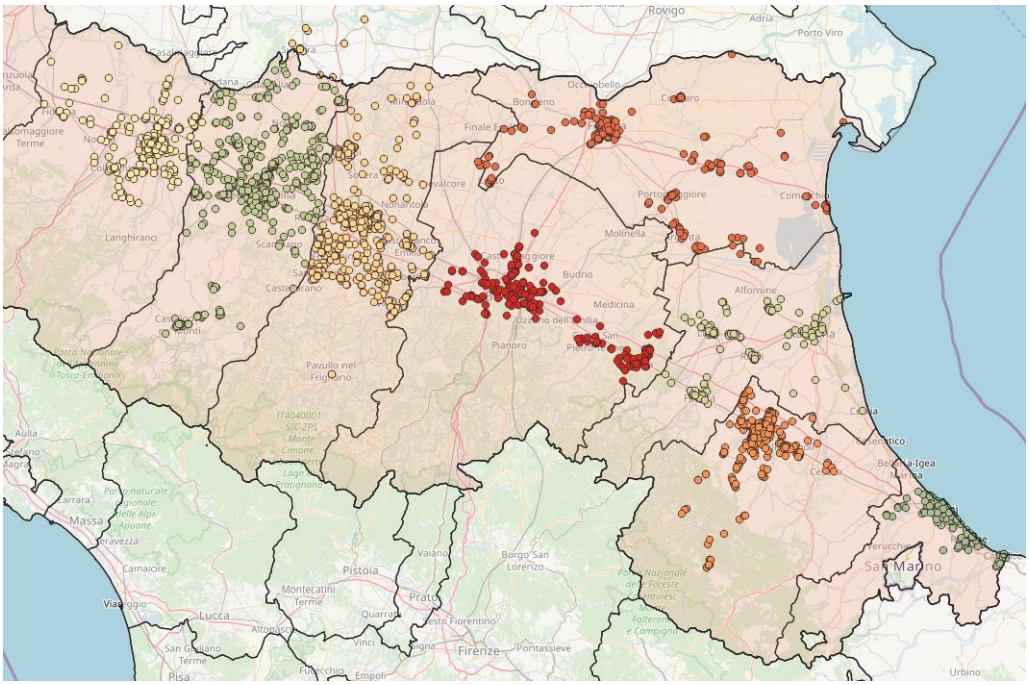


FIGURE 5.1: Customers in the Emilia Romagna region, divided by province

of Reggio Emilia, showing the customers divided into patrolling clusters. The cluster configuration does not change from a day to the other, but the routes performed inside the clusters may change according to the daily demand for services. Indeed, customers may require different services according to the day of the week, following the contract stipulated with the company. More in detail, each customer may require multiple services and, for each such service, multiple visits during the same period. Some services, such as the closing or opening of a commercial activity, are mandatory, whereas others, such as the inspection of an area or a building, are optional. The optional services induce a score when performed, and the company is interested in both maximizing the total collected score, and minimizing the total working time. As the cluster configuration is fixed, each cluster gives raise to an optimization problem that is independent from the other clusters.

The resulting optimization problem involves a number of operational constraints. First of all, the services should be performed within hard time windows and the routes should not exceed a maximum working time. In addition, a customer might require multiple visits for the same service in the same period. In such a case, two consecutive visits should be separated by at least a given threshold time (e.g., 90 minutes or so). This constraint is indeed very challenging, as it imposes to schedule endogenous time windows, induced by the consecutive visits, inside the exogenous time window imposed by the contract.

Our goal is to determine a set of routes, one per period, by optimizing an objective function that takes into account the total collected score and the total working time. The resulting problem is a multi-period orienteering problem, which is a generalization of the well-known orienteering problem (OP) (Golden, Levy, and Vohra, 1987). The OP is known to be strongly NP-hard, and difficult to solve in practice, and the problem we are facing is a challenging generalization of the OP that includes different additional constraints.

In this work, we first develop a mixed integer linear programming (MILP) model that is used to formally describe the problem and to solve some small-size instances. Since our problem is $\mathcal{NP}$-Hard, we propose a heuristic algorithm to solve large-size instances. We chose to develop an iterated local search (ILS), a metaheuristic that in recent years obtained relevant results on a large number of optimization problems Lourenço, Martin, and Stützle,

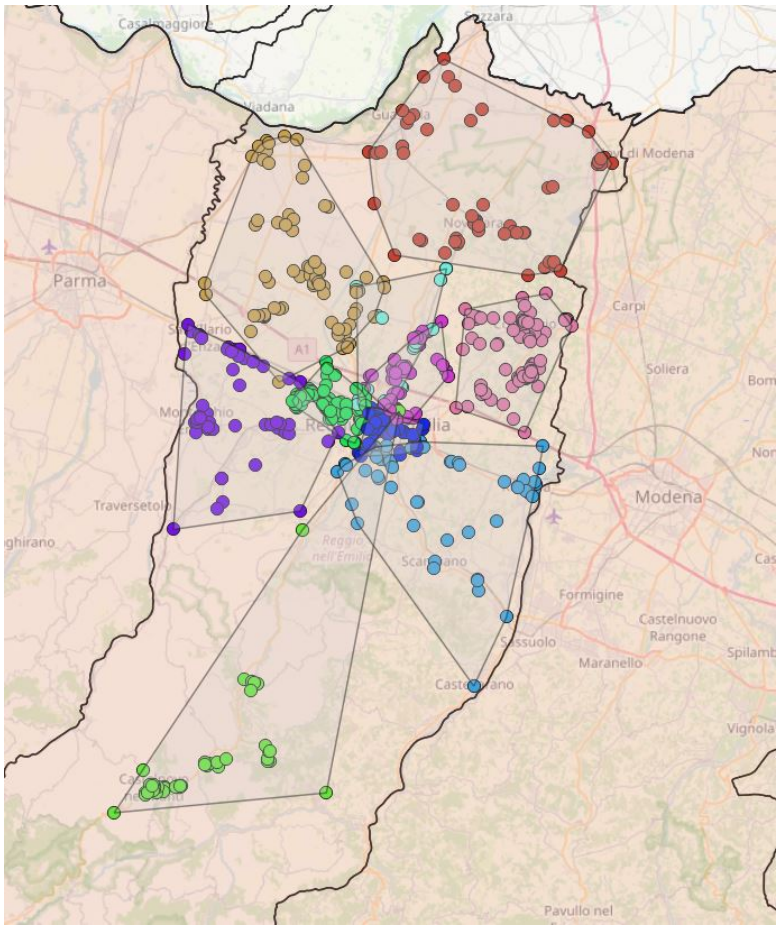


FIGURE 5.2: Customers in the Reggio Emilia province, divided by clusters

2019. The ILS receives in input the set of customers and the set of services to be performed. It first builds an initial solution by means of a constructive heuristic. Then, as long as termination conditions are not met, it iteratively applies perturbations on the current incumbent solution and looks for improvements by means of a variable neighborhood descent (VND) procedure (Hansen et al., 2019) based on six neighborhoods.

Extensive computational tests on a set of real-world instances provided by the company prove that the developed ILS works very well in practice. The solutions it obtains consistently improve both the ones produced by solving the MILP model, and the ones in use at the company, both in terms of total score and total working time.

The remainder of the paper is organized as follows. Section 5.2 contains a brief literature review of patrolling applications and OPs. Section 5.3 formally describes the problem. Section 5.4 presents the mathematical formulation. Section 5.5 gives the details of the ILS algorithm. Section 5.6 shows the computational results and, finally, Section 5.7 gives concluding remarks and hints for future research directions. A preliminary version of this work, solving a limited set of instances with just an early version of the ILS, was presented as Zucchi et al., 2022.

5.2 Brief Literature Review

Car patrolling is a security measure widely used to protect large areas from criminal activity. It consists of guards (patrols) using vehicles to move between points of interest in a region and taking actions that may prevent or respond to crimes. Car patrolling problems has been

intensively studied in the literature, sometimes under different names because they can model different applications. Very recently, Samanta, Sen, and Ghosh, 2022a; Samanta, Sen, and Ghosh, 2022b surveyed police patrolling problems, by dividing them into three categories: (i) resource allocation, (ii) district design and (iii) route design. The route design category is the one that most resembles our problem because it is concerned with how the routes are selected and how they affect the patrol efficiency. However, the problems evaluated in the survey differ considerably from ours. Indeed, whereas the police patrolling problems aim to optimize the routing coverage of an area in some ways, in our problem the patrols must visit given customers and perform pre-specified tasks related to security services.

Our optimization problem is more similar to an OP and it can be described as a generalization of a multi-period OP with time windows. The literature on OPs is very rich and one may find plenty of applications. To the best of our knowledge, the first study on the OP dates back to Tsiligirides, 1984, in which the problem was presented as a generalization of the traveling salesman problem. Because the OP is $\mathcal{NP}$-hard, most studies use heuristic methods to solve either the OP itself or some of its generalizations. A few years ago, Gendreau, Laporte, and Semet, 1998 discussed why it is so difficult to design high-quality heuristics for this class of problems. The score of a location, and the distance to reach it are independent, and often in contrast to one another, which makes it difficult to select the locations that are part of an optimal solution. For such a problem, simple construction heuristics may direct the algorithm towards undesirable directions and are not sufficient to explore large parts of the solution space. This is confirmed by the results obtained on our real-world instances, where the ILS largely outperforms the initial constructive heuristic.

In the past few years, several papers devoted to the study of OPs have been published. We refer the interested reader to the extensive reviews by Gunawan, Lau, and Vansteenwegen, 2016 and Vansteenwegen, Souffriau, and Oudheusden, 2011. More in detail, Vansteenwegen, Souffriau, and Oudheusden, 2011 formally described the most relevant problem variants, and surveyed known exact and heuristics algorithms, whereas Gunawan, Lau, and Vansteenwegen, 2016 computationally evaluated eight algorithms to solve the team orienteering problem with time windows, finding out that the ILS by Gunawan, Lau, and Lu, 2015b was the algorithm producing the best solutions on average. Surveys on related classes of problems were presented by Gavalas et al., 2014, who focused on tourist trip design problems and by Archetti, Speranza, and Vigo, 2014, who discussed vehicle routing problems with profits.

A number of papers that are closely related to our work appeared after the publication of the above surveys. A hybrid heuristic composed of a greedy randomized adaptive search procedure (GRASP) and a variable neighborhood search (VNS) was proposed by Palomo-Martínez et al., 2017 to solve a generalization of the OP. This variant contains constraints imposing mandatory visits and incompatibilities among nodes. The hybrid heuristic takes advantage of the multi-start feature of the GRASP to generate initial solutions that are then optimized with the VNS. The authors reported that the heuristic was able to find 128 optimal solutions on a set of 131 instances and required, on average, only 0.8% of the time required by an MILP model solved with a commercial solver.

The probabilistic orienteering problem is a variant of the OP in which a prize is associated with each node, but the node will be available for visit only with a certain probability. The problem has been studied by Angelelli et al., 2017, who presented an integer linear stochastic model and solved it by branch-and-cut. Computational results were presented on instances containing up 100 vertices.

A problem similar to ours, although with a different application, was studied by Kotiloglu et al., 2017 under the name of personalized multi-period tour recommendation. The goal of the problem is to generate tours including mandatory and optional visits, while maximizing the total collected score of the optional ones. The problem considers several features, such as multiple periods of visits, time windows, maximum budget and maximum tour length. The

authors presented an MILP model and an iterated tabu search. The proposed methods have been computationally evaluated by using two data sets, one from the literature and the other generated with real-world data.

The so-called Set Orienteering Problem has been studied in Archetti, Carrabs, and Cerulli, 2018. In this problem, customers are grouped in clusters and a profit is associated with each cluster, and the aim is to find a single-vehicle route that maximizes the collected profit. The authors developed a mathematical model and a matheuristic algorithm, and tested them on benchmark instances from the Generalized Traveling Salesman Problem literature involving up 1084 vertices.

In Gündling and Witzel, 2020, the goal is to optimize touristic routes considering constraints such as visit redundancy avoidance and time windows. An MILP model and an ILS metaheuristic were proposed. The authors reported that the ILS could almost match the results obtained by the MILP model solved with Gurobi for smaller instances, and for larger ones, it provided better solutions in most cases.

A multi-period orienteering problem in which a salesperson needs to perform a route to visit a subset of available customers has been studied by Zhang, Ohlmann, and Thomas, 2020. The problem is solved by a two-stage heuristic. In the first stage, the subset of customers to be visited is decided. In the second stage, a vehicle routing problem is solved by considering only the selected subset. The authors have chosen this method considering how the relationship between the customers and the salesperson works, as in their problem the salesperson has a series of decisions to make upon arriving at a customer site. The proposed method has been validated with a data set adapted from the vehicle routing problem with time windows.

Probabilistic properties in OPs have been also recently studied by Angelelli et al., 2021, who considered an online OP with stochastic service requests. Every request must be either accepted or rejected in real time, and then, at a later stage, a single vehicle must visit the accepted customers by maximizing collected profits and meet operational constraints. The authors modeled the problem as a Markov Decision Process and developed several heuristic algorithms for its solution.

In Xu et al., 2021, an approximation algorithm for a variant of the team orienteering problem (TOP) was proposed. In addition to the basic TOP constraints and the objective of maximizing the collected score, their problem includes a set of new features to better model Internet of things applications: a limited budget is imposed on the vehicles to perform the routes; node costs are included in the path cost function in addition to edge costs; nodes can be served by multiple vehicles. Computational experiments proved that the developed algorithm provided up to a 17.5% increase in the collected score compared to a state-of-the-art algorithm for the problem.

A new OP variant with service time dependent profits and time dependent travel times was investigated by Khodadadian et al., 2022. The authors proposed an MILP mathematical formulation and a VNS metaheuristic based on three specialized neighborhood structures. The authors validated their VNS on a set of benchmark instances with known optimal solutions and then they used it to solve a study case based on the city of Shiraz in Iran.

5.3 Problem description

The problem we face can be viewed as a multi-period orienteering problem with time windows (MPOPTW). In the MPOPTW, we are given a graph $G = (C_0, A)$. The set of vertices is defined as $C_0 = \{0, 1, \ldots, n\}$, where 0 is the depot at which the single vehicle starts and ends each route, and $C = \{1, \ldots, n\}$ is the set of customers. The graph is complete and a traveling time γ_{ij} is associated with each arc $(i, j) \in A$.

Let T be the set of services provided by the company. A standard service time q_t is associated with each service $t \in T$ and reports the time required by a patrol to execute such service at a customer location. The set of services is partitioned as $T = M \cup U$, where M is the set of mandatory services and U is the set of optional ones. The activities should be executed on a given set D of periods. Each period $d \in D$ corresponds to a working shift of a patrol, with a given start and maximum end time. Each customer $c \in C$ requires services on a subset $D_c \subseteq D$ of periods. Formally, we denote by $T_{cd} \subseteq T$ the set of services to be performed at customer c on period d. This set is partitioned as $T_{cd} = M_{cd} \cup U_{cd}$, where $M_{cd} \subseteq M$ comprises mandatory services and $U_{cd} \subseteq U$ optional ones.

Let n_{cdt} be the number of times service t is required by customer c in period d and let $\bar{n}_{cdt}$ be the number of services that have been actually performed in a solution. We define the quality of service (QoS) level as $Q = \sum_{c \in C, d \in D_c, t \in T_{cd}} \bar{n}_{cdt} \,/\, \sum_{c \in C, d \in D_c, t \in T_{cd}} n_{cdt}$. Index Q represents the ratio of services that have been performed in the entire set of periods and it should be greater than or equal to an input threshold value $Q_{\min}$.

Every service t required by a customer c in a period d is associated with a time window $[e_{cdt}, l_{cdt}]$. This defines the earliest and latest possible times to start the execution of each of the n_{cdt} services. The time window defines a hard constraint: late arrivals are forbidden and waiting on site is imposed in case of early arrivals. A time window $[e_0, l_0]$ is also imposed on the depot and sets the maximum working time in a period (from 22:00 of a day to 06:00 of the next day in our instances). For some services, such as the closing or the opening of a commercial activity, the time window is strict (e.g., 10 minutes) and just one visit per night is required. This is typically the case for mandatory services. For other services, such as checking a private house, the time window is usually loose (e.g., several hours) but multiple visits may be required in a period. This is typically the case for optional services. In such a case, if two or more visits are performed for the same service at the same customer in the same period, then the start times of any two of such visits should be separated by at least a given threshold $\delta_{\min}$ (which is equal to 90 minutes in our instances). This is imposed to enforce a balanced patrol of the customer during the execution of a route.

For each period, a patrol starts its route at the depot, performs visits to customers to execute the services and then returns to the depot. The working time of a route is defined as the difference between the time at which the vehicle returns to the depot and the beginning of the shift. The beginning of the shift is e_0 and the route working time cannot exceed the maximum duration defined by $l_0 - e_0$.

Each service $t \in T$ required by a customer $c \in C$ is associated with a score w_{ct}. This score is collected during the first time the service is performed at the customer in a given period, and it does not vary from one period to the other. If multiple visits are performed in one period for the same service at the same customer, then the collected additional score varies according to a decreasing function. In detail, let $\tau = 1, \ldots, n_{cdt}$ be the index of the τth visit performed at customer c for service t in period d. Then, the score collected at visit τ is $w_{ct\tau}$ and is such that $w_{ct1} = w_{ct}$ and $w_{ct\tau} \geq w_{ct,\tau+1}$ for $\tau = 1, \ldots, n_{cdt} - 1$. In this way, the more visits for a given service are performed at a customer in a period, the more the score decreases and hence the first visits for other services and/or other customers become preferable to another visit for the same service at the same customer. This helps to achieve a balanced number of visits among customers and services.

To summarize, the aim of the MPOPTW is to define a set of routes, one per period, in such a way that (i) all mandatory services are performed, (ii) all operational constraints are satisfied, and (iii) a weighted function, which considers the score $\mathcal{S}$ of the services that have been actually performed, and the total working time $\mathcal{T}$ (with a negative weight on $\mathcal{T}$), is maximized. Two input parameters, α and β, are used as weights of $\mathcal{S}$ and $\mathcal{T}$, respectively, and the function to be maximized is $z = \alpha\mathcal{S} - \beta\mathcal{T}$.

5.4 Mathematical Model

To model the MPOPTW as an MILP, we work on an extended graph in which each vertex is used to represent a visit to a customer to perform a service. In detail, let V_d be the set of vertices representing all possible visits associated with the services requested by all customers in period $d \in D$. Let $n = |\cup_{d\in D} V_d|$ be the total number of vertices and note that by construction $n = \sum_{c\in C, d\in D, t\in T} n_{cdt}$. Let also $V_d^0 = V_d \cup \{0\}$ and $V_d^{n+1} = V_d \cup \{n+1\}$, where 0 and $n+1$ are copies of the depot representing, respectively, the start and end of the route for each period $d \in D$.

We define V_{cdt} as the subset of V_d representing the n_{cdt} visits for service $t \in T$ requested by customer $c \in C$ in period $d \in D$. Each vertex $v \in V_{cdt}$ is associated with the standard service time and the time window associated with the corresponding service t and customer c, respectively. Similarly, the traveling time between two vertices is set to be equal to the traveling time between the two customers associated with the vertices.

For what concerns the scores, we use w_{vd} to denote the score associated with vertex $v \in V_d$ in period $d \in D$. The value of w_{vd} is equal to the score associated with the corresponding visit. It is important to notice that vertices in each subset V_{cdt} are sorted by decreasing score. That is, the first vertex in V_{cdt} corresponds to the first visit to perform service t at customer c in period d and hence has the highest score, the second vertex corresponds to the second visit and hence has the second highest score, and so forth, for each $c \in C$, $t \in T$, and $d \in D$.

Three sets of decision variables are defined: (i) x_{ijd} takes the value 1 if the patrol moves from vertex $i \in V_d^0$ to vertex $j \in V_d^{n+1}$ in period $d \in D$, 0 otherwise; (ii) y_{vd} takes the value 1 if vertex $v \in V_d$ is visited in period $d \in D$; (iii) s_{vd} gives the time at which the patrol arrives at vertex $v \in V_d^{n+1}$ in period $d \in D$.

The MILP model is defined as follows:

$$\max z = \quad \alpha \sum_{d\in D}\sum_{v\in V_d} w_{vd} y_{vd} - \beta \sum_{d\in D} s_{n+1,d} \tag{5.1}$$

$$\text{s. t.} \quad \sum_{j\in V_d^{n+1}} x_{0jd} = \sum_{i\in V_d^0} x_{i,n+1,d} = y_{0d} = y_{n+1,d} = 1 \qquad d \in D \tag{5.2}$$

$$\sum_{i\in V_d^0} x_{ivd} = \sum_{j\in V_d^{n+1}} x_{vjd} = y_{vd} \qquad v \in V_d, d \in D \tag{5.3}$$

$$\sum_{v\in V_{cdt}} y_{vd} = n_{cdt} \qquad c \in C, d \in D_c, t \in M_{cd} \tag{5.4}$$

$$s_{id} + q_i + \gamma_{ij} - \mathcal{M}(1 - x_{ijd}) \le s_{jd} \qquad i \in V_d^0, j \in V_d^{n+1}, d \in D \tag{5.5}$$

$$e_{id} - \mathcal{M}(1 - y_{id}) \le s_{id} \qquad i \in V_d^0, d \in D \tag{5.6}$$

$$s_{jd} \le l_{jd} + \mathcal{M}(1 - y_{jd}) \qquad j \in V_d^{n+1}, d \in D \tag{5.7}$$

$$s_{jd} - s_{id} \ge \delta_{\min} - \mathcal{M}(2 - y_{id} - y_{jd}) \qquad i, j \in V_{cdt} : i < j, t \in T_{cd}, c \in C, d \in D \tag{5.8}$$

$$\frac{1}{n}\sum_{d\in D}\sum_{v\in V_d} y_{vd} \ge Q_{\min} \tag{5.9}$$

$$x_{ijd} \in \{0,1\} \qquad i \in V_d^0, j \in V_d^{n+1}, d \in D \tag{5.10}$$

$$y_{vd} \in \{0,1\} \qquad v \in V_d \cup \{0, n+1\}, d \in D \tag{5.11}$$

$$s_{id} \ge 0 \qquad i \in V_d \cup \{0, n+1\}, d \in D. \tag{5.12}$$

The objective function (5.1) maximizes the weighted sum of total collected score minus total working time, multiplied by, respectively, α and β. Constraints (5.2) guarantee that each route starts and finishes at the depot. Constraints (5.3) guarantee route connectivity and also enforce the relation between variables x and y. Constraints (5.4) guarantee that all mandatory

services are executed. Constraints (5.5), (5.6) and (5.7) enforce the relation between variables x and s, and they also impose time window constraints on each service that is executed. In these constraints, $\mathcal{M}$ is used to denote a large number. Constraints (5.8) guarantee that visits for the same service required by a customer are separated by at least $\delta_{\min}$ units of time. Constraint (5.9) imposes the minimum QoS. Constraints (5.10), (5.11) and (5.12) give the domain of the decision variables.

5.5 Iterated Local Search

To solve large-size MPOPTW instances, we have developed an ILS metaheuristic. The ILS builds an initial solution by using a constructive heuristic and then iteratively applies perturbations and local searches over the current solution until a termination condition is reached. The perturbation step accepts only solutions that are feasible with respect to all constraints of the problem, including the minimum QoS. The local search is performed by means of a VND, an algorithm that sequentially invokes local search procedures with a set of neighborhoods and finds a locally optimal solution for this set (Hansen et al., 2019). An acceptance function decides at each iteration whether to keep the current solution or move to a newly-generated one. In our case, the newly-generated solution is accepted only if its value is better than that of the current solution. The overall ILS procedure is presented in Algorithm 8. The algorithm runs until either a maximum run time or a maximum number of iterations without improvements is reached. Each step of the ILS is described in detail in the following.

Algorithm 8 ILS algorithm

```
1: procedure ILS(T_max = maximum run time, I_max = maximum number of iterations without improvements)
2:     ITERATION ← 0                          ▷ the number of iterations without improvement
3:     s* ← CONSTRUCTIVEHEURISTIC
4:     s* ← VND(s*)
5:     while ELAPSEDTIME ≤ T_max or ITERATION ≤ I_max do
6:         s' ← PERTURBATION(s*)
7:         s'' ← VND(s')
8:         if ACCEPT(s*, s'') then
9:             s* ← s''
10:            ITERATION ← 0
11:        else
12:            ITERATION ← ITERATION + 1
13:        end if
14:    end while
15:    return s*
16: end procedure
```

Evaluation function. Our ILS uses the objective function as it is to evaluate solutions. Let $\mathcal{S}(\sigma_d)$ be the total score of a given route σ_d performed in period d and $\mathcal{T}(\sigma_d)$ be its working time. Then the objective function of a solution $s = \{\sigma_d : d \in D\}$ can be expressed as follows:

$$z(s) = \alpha \sum_{d \in D} \mathcal{S}(\sigma_d) - \beta \sum_{d \in D} \mathcal{T}(\sigma_d). \tag{5.13}$$

Constructive heuristic. An initial solution is constructed by a greedy algorithm, whose pseudo-code is summarized in Algorithm 9. The vertices of each period are sorted in non-decreasing order of the start time of their time windows. A route is constructed for each period in two phases: first, the mandatory vertices are inserted sequentially, each at the end of the current route, in the order in which they were sorted provided that this preserves

feasibility, that is, a vertex is appended only if the solution remains feasible, otherwise it is skipped (this is always feasible for the mandatory services in the instances provided by the company); later, optional vertices are appended one by one in the solution in the sorted order, each at the end of the current route, whenever the resulting route is feasible. The two phases invoke the procedure Append(σ_d, v) that first checks if inserting vertex v at the end of route σ_d is feasible, and then, if feasibility is confirmed, it returns the expanded route having v at its end; otherwise, it return the original route σ_d.

Algorithm 9 The greedy constructive heuristic

```
1:  procedure CONSTRUCTIVEHEURISTIC
2:      for each period d ∈ D do
3:          M_d, U_d ← services in M and U for period d
4:          Sort M_d and U_d in non-decreasing order of e_vdt
5:          σ_d ← ∅                                          ▷ empty route
6:          for v ∈ M_d (in the sorted order) do
7:              σ_d ← Append(σ_d, v)          ▷ append mandatory v at the end if feasible
8:          end for
9:          for v ∈ U_d (in the sorted order) do
10:             σ_d ← Append(σ_d, v)           ▷ append optional v at the end if feasible
11:         end for
12:         end for
13:         return s = {σ_1, ..., σ_D}
14:     end procedure
```

Variable Neighborhood Descent. A VND procedure is used to find a locally optimal solution using a sequence of different neighborhoods N_k ($k = 1,\ldots,k_{\max}$). Algorithm 10 shows its main steps. Starting with the first neighborhood ($k = 1$), the VND explores the solution space by searching through the sequence of neighborhoods in a deterministic way. More in detail, at each step of the VND, the current solution is brought to a locally optimal solution by exploring the current neighborhood N_k using the first improvement policy. If no solution better than the current one is found in the kth neighborhood, then the algorithm switches to the next neighborhood, N_{k+1}. If, instead, a better neighbor solution is found, then this solution is used to replace the current one and the algorithm returns to the first neighborhood, N_1. The process continues while there is a neighborhood to be explored, that is, it stops when the current solution is locally optimal with respect to all neighborhoods.

Algorithm 10 Variable neighborhood descent heuristic

```
1:  procedure VND(s)
2:      k ← 1
3:      while k ≤ k_max do
4:          s' ← HillClimbing(s, N_k)          ▷ Find a locally optimal solution
5:          if z(s') > z(s) then                          ▷ Neighborhood change
6:              s ← s'
7:              k ← 1
8:          else
9:              k ← k + 1
10:         end if
11:     end while
12:     return s
13: end procedure
```

We implemented six neighborhoods. Some of them are classical neighborhoods from the vehicle routing literature, whereas others were specifically designed to meet our problem requirements. The neighborhoods are as follows:

- N_1 = Remove optional: Remove an optional service vertex from a route, thus trying to decrease the working time;
- N_2 = Swap: Swap the positions of two vertices inside a route;
- N_3 = 2-opt: Swap two arcs in a route, reversing the visiting order between the two arcs;
- N_4 = Relocate: Move a vertex to another position in the route;
- N_5 = Insert optional: Insert an optional unvisited vertex into a route, in an attempt to increase the collected score;
- N_6 = Swap optional: Swap two optional vertices, by letting an unvisited vertex take the place of a visited one.

The neighborhoods Swap, 2-opt and Relocate may improve the objective function only by reducing the working time of a route, because they do not change the collected score. The Remove optional movement attempts to decrease the working time at the expense of a decrease in the score as well. On the contrary, the Insert optional neighborhood tries to improve the score at the expense of an increase in the working time. The last neighborhood, Swap optional, may improve the objective function by increasing the score, reducing the working time, or both.

Note that all described neighborhoods consist of intra-period and intra-route movements, as they change routes of each period independently. A solution may be further improved by performing inter-period movements, that is, changing the execution of a service at a customer from a period to another. This type of movements may decrease the working time in a period and open space for more services to be performed there. Inter-period movements are costly to be evaluated because of the large number of neighbors. Thus, they are not fully explored in a deterministic way, but are considered in the perturbation step described next.

Perturbation procedure. The perturbation procedure is introduced to escape from the locally optimal solution obtained by the VND. Two inter-period neighborhoods are used to this aim. At each iteration, the perturbation procedure randomly selects two periods, d_1 and d_2, and then it invokes, alternatively, one of the two neighborhoods. The Relocate inter-period neighborhood randomly selects a vertex i_1 in σ_{d_1}, removes it from σ_{d_1} and tries to insert it in every position of σ_{d_2} (provided that there is a demand left for i_1 in d_2). Similarly, the Swap inter-period neighborhood randomly chooses a vertex i_1 in σ_{d_1} and then it tries to swap it with any other vertex i_2 in σ_{d_2}. In either case, if a move succeeds in producing a feasible solution, then it is applied, independently of the cost; otherwise, it is rejected. Either neighborhood proceeds until 10 successful moves have been produced, or $|\sigma_{d_1}||\sigma_{d_2}|$ attempts (either successful or unsuccessful) have been performed, where $|\sigma_d|$ signifies the number of vertices in σ_d

Figure 5.3 illustrates the Relocate inter-period operation through a simplified example. Three periods, namely 1, 2 and 3, are considered. The top part of the figure illustrates the solution before the operation is applied and the bottom part illustrates it after. All routes start from the depot and visit customer $c = 1$ to provide different services. Suppose the customer requires services 1, 2, 3 and 4 in periods 1 and 2, and services 1, 2 and 3 in period 3. Suppose also that a single visit per service, and per period is required, and that service 2 is optional. The depicted move removes the execution of service 2 in period 1 and transfers it to period 3. In this way, the route in period 1, which had the service removed, may be used to include other services.

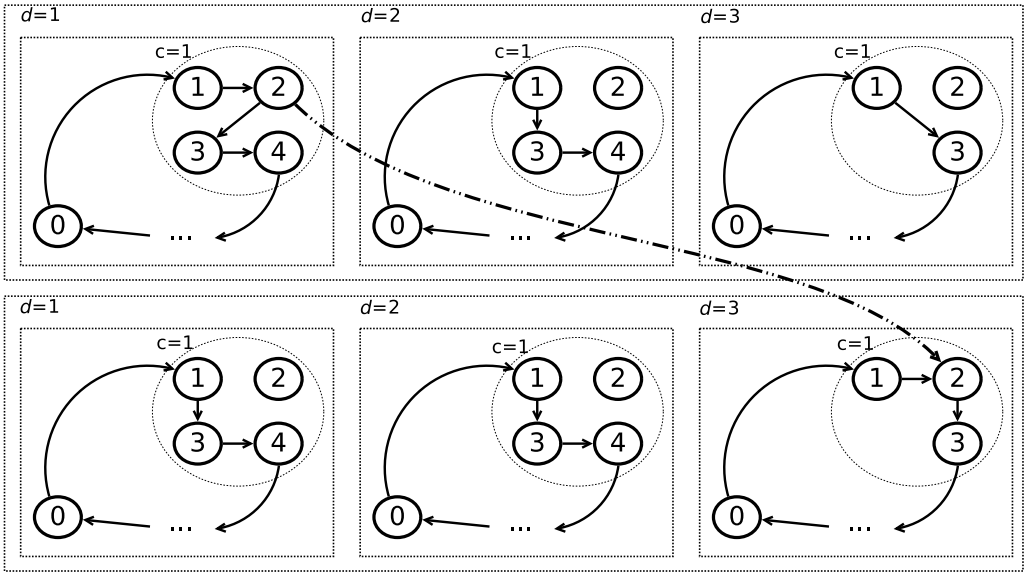


FIGURE 5.3: Depicting example for the Relocate inter-period perturbation movement

5.6 Computational evaluation

In this section, we present the outcome of the extensive computational tests that we performed on a large set of real-world instances provided by the company. The algorithms have been coded in Python 3.7.3 and the MILP model was solved with Gurobi 9.5. Gurobi was invoked with its default configuration, letting it run on 12 threads. The experiments have been executed on a Intel Xeon CPU E5-2640 v3 2.60 GHz machine with 64 GB of memory, running under Windows 10 Pro 20H2 64-bits.

5.6.1 Instances

The company provides security services in a number of provinces in Italy. We were provided with the data of 12 of such provinces, which differ among them in the number and geographical distribution of customers, as well as in the number of services requested. Table 5.1 reports for each province, in order, the number of instances (column #), which also corresponds to the number of clusters, the number of periods ($|D|$), the total, average, minimum, and maximum number of customers ($\text{tot}_{|C|}$, $\text{avg}_{|C|}$, $\text{min}_{|C|}$ and $\text{max}_{|C|}$), and of services requested (tot_n, avg_n, min_n and max_n). We were provided in total with 79 instances, with the number of customers varying from 5 to 100 and the number of requested services from 14 to 1280.

The values of α and β, used in the objective function, were provided by the company after internal discussion and were set to 5 and 0.9, respectively. The score collected during visit $\tau = 1,\ldots,n_{cdt}$ performed at customer c for service t in a period was set to $w_{ct\tau} = w_{ct}e^{1-\tau}$, with the w_{ct} values being in the set $\{1,6,9,10\}$. The minimum QoS level has been set to 75% and the minimum time between two consecutive visits for the same service at a customer to $\delta_{\min} = 90$ minutes. The traveling times have been obtained by computing the real-world distances using the Open Source Routing Machine application.

TABLE 5.1: Details of the real-world instances

Province	#	$\|D\|$	$tot_{\|C\|}$	$avg_{\|C\|}$	$max_{\|C\|}$	$min_{\|C\|}$	tot_n	avg_n	max_n	min_n
Mantova	2	7	43	21.5	32	11	171	85.5	140	31
Roma	8	7	118	14.8	25	5	1234	154.3	268	84
Sassari	7	7	119	17.0	33	7	1038	148.3	256	42
Rimini	4	7	154	38.5	57	27	987	246.8	415	93
Ravenna	5	7	172	34.4	50	15	1382	276.4	400	79
Pescara	4	7	227	56.8	69	43	1370	342.5	484	129
Ferrara	7	7	236	33.7	63	11	1812	258.9	525	97
Bologna	8	7	239	29.9	63	17	2733	341.6	632	244
Parma	5	7	289	57.8	77	37	2952	590.4	797	323
Forli	8	7	407	50.9	73	16	2695	336.9	586	40
Modena	11	7	510	46.4	76	9	4288	389.8	651	14
Reggio Emilia	10	7	679	67.9	100	22	7812	781.2	1280	243

5.6.2 ILS results

Table 5.2 reports the results obtained by the ILS. Due to the high number of instances, we chose to aggregate the results by province. For each province we provide several key performance indicators (KPI): the average objective function value, z, according to Equation (5.1); the average collected score, $\mathcal{S}$; the average working time, $\mathcal{T}$; the average traveled distance by a patrol, km; the average time between two consecutive visits at the same customer to perform the same service, δ; the average quality of service, Q; the average time in which the incumbent solution was found, time_{inc}; and the average run time in seconds, time. We also include columns #, $avg_{|C|}$ and avg_n for the sake of easy data visualization. The last line provides overall averages.

TABLE 5.2: Average computational results obtained by the ILS

Instance				ILS							
Province	#	$avg_{\|C\|}$	avg_n	z	$\mathcal{S}$	$\mathcal{T}$	km	δ	Q	time_{inc}	time
Mantova	2	21.5	85.5	2164.7	527.6	526.1	75.0	92.3	82.6	58.1	99.9
Roma	8	14.8	154.3	2945.6	828.5	1329.9	145.2	98.9	82.2	117.8	187.9
Sassari	7	17.0	148.3	2941.7	762.3	966.3	105.4	94.4	89.1	99.8	178.8
Rimini	4	38.5	246.8	5093.6	1168.6	832.8	84.6	91.5	92.2	870.6	1250.3
Ravenna	5	34.4	276.4	6629.6	1547.4	1230.2	110.0	124.4	95.7	823.1	1488.6
Pescara	4	56.8	342.5	9396.1	2102.5	1240.3	156.4	115.6	91.8	1223.7	1858.7
Ferrara	7	33.7	258.9	7162.0	1669.6	1317.5	136.4	104.2	96.5	692.9	797.4
Bologna	8	29.9	341.6	8217.4	2051.8	2268.3	184.1	125.6	94.9	792.4	944.8
Parma	5	57.8	590.4	15593.8	3441.8	1794.5	162.8	131.7	93.8	3101.5	3466.2
Forli	8	50.9	336.9	9159.1	2034.8	1127.9	126.9	110.8	97.8	1042.4	1400.5
Modena	11	46.4	389.8	9380.1	2234.0	1988.5	220.5	148.9	96.7	873.2	1243.9
Reggio Emilia	10	67.9	781.2	16956.2	3818.8	2375.5	220.2	146.1	94.5	2623.4	3024.9
Average	6.6	40.4	360.4	8600.7	2002.4	1568.3	157.6	119.8	93.1	1077.5	1372.7

The ILS was able to find a feasible solution for every instance in an average time of about 23 minutes. The average time was around 3 minutes or less for Mantova, Roma and Sassari, provinces requiring a small number of services, and larger than 50 minutes for Reggio Emilia, the province requiring the largest number of services. By comparing time_{inc} with time, we observe that the criteria adopted for the ILS termination are appropriate because the algorithm keeps running for some amount of time after the incumbent has been found, but this time is not excessive. For all instances, the constraints on the minimum QoS and minimum time between services were satisfied, and the average Q and δ values are far above the required 75% and 90 minutes, respectively. In the next sections, we obtain insights in the remaining

KPIs by comparing them with the corresponding values obtained by the MILP model and by the company.

5.6.3 Comparison with the mathematical model

Table 5.3 reports the results obtained by solving the MILP model with Gurobi and compares them with the results by the ILS. Let F be the set of instances for which Gurobi was able to find a feasible solution. Column $|F|$ shows the number of such instances in F for each province (e.g., Gurobi obtained feasible solutions for 2 out of 2 instances of Mantova, 8 out of 8 instances of Roma, 6 out of 7 instances of Sassari). The next columns refer to average values with respect to this reduced set of instances. Columns *LB*, *UB*, gap and time provide the average incumbent solution value, the average upper bound, the average percentage gap between *LB* and *UB*, and the average run time, respectively, of Gurobi. For the ILS, we report z, time, and, in column impr., the average improvement obtained over the incumbent values *LB* of Gurobi, defined as $100(z-LB)/LB$. The values of the ILS reported in this table were also taken for the subset F of instances.

TABLE 5.3: Comparison between Gurobi and ILS (on subset F of instances for which MILP found a feasible solution)

Instance		MILP Model					ILS		
Province	#	$\|F\|$	*LB*	*UB*	gap	time	z	impr.	time
Mantova	2	2	2134.9	2754.6	22%	3600.1	2164.7	1.5%	99.9
Roma	8	8	2828.9	4104.0	31%	3600.1	2945.6	12.6%	187.9
Sassari	7	6	2755.8	3531.7	22%	3600.1	2809.7	0.9%	168.4
Rimini	4	4	4763.7	5906.5	19%	3600.3	5093.6	5.0%	1250.3
Ravenna	5	3	5018.2	6220.7	19%	3600.4	5403.3	5.2%	946.7
Pescara	4	2	6781.9	8030.9	16%	3600.2	7118.4	3.9%	679.3
Ferrara	7	4	4989.5	6080.5	18%	3600.2	5094.2	1.6%	79.2
Bologna	8	2	5419.1	7394.3	27%	3600.2	6191.9	18.6%	970.3
Parma	5	0				3600.1			
Forli	8	4	6817.5	7751.7	12%	3600.3	7134.7	3.6%	942.5
Modena	11	1	321.9	321.9	0%	0.8	321.9	0.0%	0.1
Reggio Emilia	10	1	3976.3	5865.8	32%	3600.4	4669.0	17.4%	775.7
Average	6.6	3.1	4147.9	5248.3	22%	3502.9	4374.7	6.2%	505.9

Only in three cases, the MILP solver could find feasible solutions for all instances of a province. This happened for Mantova, Roma and Rimini. Moreover, the solver was not able to find any solution for the instances of Parma, and just a single one for the instances in Modena and Reggio Emilia. In total, only 37 out of 79 instances were feasibly solved and just one of them (Modena) to proven optimality. On this subset F of instances, the ILS was able to improve the solution value found by the MILP solver by 6.2% on average. The run time was also considerably lower. Whereas Gurobi reached the time limit of one hour in most cases, the ILS required on average about eight minutes and a half. We also notice that the ILS was able to find an exact optimal solution for the only instance that Gurobi could solve to proven optimality.

In Figure 5.4, we gain further insight by displaying four KPIs derived from the solutions obtained by the MILP solver and by the ILS. We still consider only the subset F of instances, providing the average score, working time, QoS and km traveled. For five provinces, namely Mantova, Roma, Sassari, Rimini and Ferrara, the solver was able to obtain a slightly better score than the ILS. On the other cases, the ILS got better or equal average scores, achieving an overall advantage of 0.32% better average score than Gurobi. Whereas the differences in the score are not so significant, much higher differences can be observed for the working

time. The ILS found, indeed, better values than Gurobi for the instances of nine provinces and an identical value for the remaining instance (Modena). The average QoS produced by Gurobi is 91.4% and that of the ILS is just 93.1%. Both values are thus much higher than the minimum required QoS (75%). Significant gains are obtained by the ILS for the km traveled for all instances with the exception of Modena (equal values) and Sassari (slightly worse value by the ILS).

(A) Average score ($\mathcal{S}$)

(B) Average working time ($\mathcal{T}$)

(C) Average quality of service (Q)

(D) Average traveled kilometers (km)

FIGURE 5.4: Four KPIs by Gurobi and ILS (on subset F of instances for which MILP found a solution)

5.6.4 Comparison with the company solutions

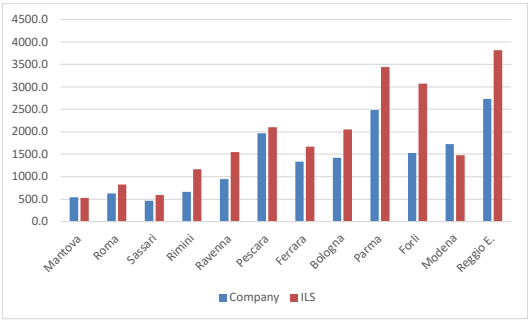
In Table 5.4, we compare the ILS solutions with the real-world ones currently in use at the company. We display the average solution value, QoS and δ for both solution sets. For the ILS, we also report the percentage improvement that it obtained over the z value by the company.

The solutions currently in use at the company are difficult to evaluate because they do not obey at least two constraints. First, they do not ensure a minimum quality of service. Refer for example to the results for the province of Pescara, where the QoS is 48.6% on average, whereas the minimum desired is 75%. In addition, the minimum time between two visits at the same customer to perform the same service is not fully respected. The average δ on the solutions provided by the company is indeed 58.4 minutes, whereas the minimum required is 90 minutes. Thus, the solutions by the company are frequently infeasible, whereas the ones produced by the ILS are all feasible. Even in this disadvantageous scenario, the ILS found solutions whose average value is much better than that of the company.

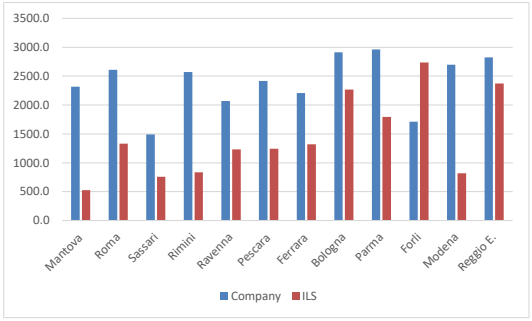
We graphically compare in Figure 5.5 two KPIs, score and working time, to show the improvements obtained by the ILS. The collected score is improved in all provinces, with the exceptions of Mantova (2% worse) and Modena (17% worse). The working time is reduced in all provinces, with the exception of Forli. Some reductions are significant as, for example, in Mantova, Modena and Rimini.

TABLE 5.4: Comparison between company and ILS solutions (on the entire set of instances)

Instance		Company			ILS			
Province	#	z	Q	δ	z	Q	δ	impr.
Mantova	2	614.7	80.6	41.7	2164.7	82.6	92.3	252%
Roma	8	788.6	83.1	87.7	2945.6	82.2	98.9	274%
Sassari	7	1002.9	92.7	45.4	2941.7	85.9	94.4	193%
Rimini	4	1024.4	78.4	16.4	5093.6	92.2	91.5	397%
Ravenna	5	2884.7	90.8	51.9	6629.6	95.7	124.4	130%
Pescara	4	7657.0	48.6	30.7	9396.1	91.8	115.6	23%
Ferrara	7	4700.0	96.7	60.7	7162.0	96.5	104.2	52%
Bologna	8	4475.1	98.5	77.5	8217.4	94.9	125.6	84%
Parma	5	9767.1	92.6	85.4	15593.8	93.8	131.7	60%
Forli	8	6210.2	83.7	63.9	9159.1	97.8	110.8	47%
Modena	11	6115.9	75.4	36.3	9380.1	96.7	148.9	53%
Reggio Emilia	10	11114.5	83.5	68.3	16956.2	94.5	146.1	53%
Average	6.6	5181.6	84.8	58.4	8600.7	92.8	119.8	66%



(A) Average score ($\overline{s}$)



(B) Average working time ($\overline{T}$)

FIGURE 5.5: Relevant KPIs for company and ILS (on the entire set of instances)

5.6.5 Evaluation of the ILS components

The proposed ILS contains several components. To evaluate the contribution of each one of them, we executed different configurations of the ILS, each obtained by removing one or more components. The obtained results are shown in Table 5.5, which displays the average solution values per province for several tested configurations. In column Greedy, we show the results of the greedy constructive heuristic alone, and in column G. + VND, the results of the greedy followed by the first VND execution. The next six columns show the results of the complete ILS, but removing one of the six VND neighborhoods. Finally, for the purpose of comparison, the last column displays the results of the complete ILS with all neighborhoods, as previously reported in Table 5.2.

We can notice that the greedy algorithm provides low-quality solutions in general, which are very far away from the final solutions obtained by the ILS. The VND (with all neighborhood searches) manages to consistently improve the solutions by the greedy. A large improvement can be observed for the eight instances of Bologna, for which the average objective value is almost doubled.

For what concerns the different neighborhood searches, we can notice that all of them

TABLE 5.5: Impact of the ILS components on the solution value

Instances		Heuristics								
Province	#	Greedy	G. + VND	ILS - N_1	ILS - N_2	ILS - N_3	ILS - N_4	ILS - N_5	ILS - N_6	ILS
Mantova	2	1787.4	2149.0	1897.2	2164.3	2163.7	2158.4	2160.5	2164.9	2164.7
Roma	8	1965.6	2924.6	2245.5	2946.3	2931.4	2922.5	2907.7	2936.6	2945.6
Sassari	7	2243.8	2923.1	2565.1	2931.7	2939.3	2929.8	2940.6	2931.5	2941.7
Rimini	4	4216.5	4872.3	4820.2	5011.7	4998.9	5047.9	4924.9	5004.3	5093.6
Ravenna	5	5634.1	6314.9	6439.8	6540.2	6418.9	6586.9	6358.5	6340.5	6629.6
Pescara	4	7717.6	8942.9	8641.5	9165.7	9022.8	9131.1	8574.5	8953.6	9396.1
Ferrara	7	5942.6	6986.9	6944.8	7008.6	6978.4	7033.8	6769.7	7001.4	7162.0
Bologna	8	4168.8	8166.3	8053.9	8208.0	8160.1	8184.4	4967.6	8203.0	8217.4
Parma	5	11177.2	13452.9	13582.5	14580.5	13938.4	14919.2	12627.5	13616.3	15593.8
Forli	8	7135.9	9031.3	9006.3	9079.3	9045.8	9116.4	7951.8	9055.3	9159.1
Modena	11	2927.1	9335.2	9338.6	9346.6	9324.2	9344.2	3618.7	9340.3	9380.1
Reggio Emilia	10	8346.2	13374.2	13597.7	14420.7	14141.6	14833.3	9899.7	13851.8	16956.2
Average		5246.8	7913.6	7816.1	8166.0	8059.6	8243.2	6160.0	8003.7	8600.7
ILS improvement		64%	9%	10%	5%	7%	4%	40%	7%	-

have a positive impact on the performance of the ILS. Removing N_1 (Remove optional) leads to an average z equal to 7816.1, which is even lower than the average values found by greedy + VND with all neighborhoods (7913.6). A smaller impact can be observed by the removals of N_2, N_3, N_4 and N_6, which lead to average solution values 5%, 7%, 4% and 7%, respectively, away from the one found by the full ILS. The largest deterioration was observed when removing N_5 (Insert optional). In this case, the average z value is 40% away from the one by the full ILS, and the distance is very large for the most difficult provinces, such as Modena (3618.7 vs 9380.1) and Reggio Emilia (9899.7 vs 16956.2).

We can conclude that all neighborhoods are important and that the two most indispensable ones are, in order, Insert optional and Remove optional. The importance of Insert optional follows from the fact that the greedy algorithm is quite inefficient in including optional services in the routes; hence the neighborhood can insert many of such services and consistently improve the solution value. The Remove optional is an important operation because it opens the possibility for the VND to decrease the collected score but at the same time decrease the total working time. In this way, the algorithm has more chances to escape from locally optimal solutions.

5.7 Conclusions

We studied a car patrolling application that arises from a large Italian company that needs to plan routes to perform security services at customers' facilities. The resulting optimization problem is a challenging variant of the multi-period orienteering problem. Due to the difficulty of the problem, we have chosen to solve it through an ILS equipped with an inner VND. We have also developed an MILP model to formalize the problem.

We have tested both the MILP model, using the Gurobi solver, and the ILS on real-world instances provided by the company. Gurobi struggled to provide feasible solutions for about half of the instances, whereas the ILS could solve all of them. Comparing only the subset of instances in which both the ILS and the solver were able to find feasible solutions, we noticed that the ILS could provide much better solution quality within a smaller computational effort. The qualities of service obtained by the two methods were approximately the same, but the ILS was able to considerably reduce the kilometers travelled by the patrols. With our tests, we thus ensured that the ILS is a preferable choice to solve the problem.

We then compared the solutions obtained by the ILS with those in use at the company. It was difficult to fairly compare their objective function values because many of the solutions in use at the company did not respect some of the operational constraints (minimum time between consecutive visits and minimum QoS). The ILS was still able to find solutions with

better objective values while respecting all operational constraints. The average improvement in the objective function value was 47.3%, and not only the minimum QoS was always respected, but it was also increased on average by about 10%.

These good results have been obtained by a simple but effective combination of several algorithmic components, including a local search algorithm with six different neighborhoods. By performing a sensitivity analysis on the entire set of instances, we observed that all such neighborhoods have a positive contribution to the ILS performance. The largest contributions are provided, in order, by Insert optional and Remove optional, two neighborhood operations that are tailored to the problem at hand.

For future research directions, we consider the following worth investigating. First, the modification of the clusters: the current clusters were provided by the company, but we foresee that changing their configuration might help improve the solution quality even further. Second, a more elaborated evaluation of the quality of service: in our study, the quality of service is evaluated using an overall measure of the number of services performed, which may introduce unfairness because it is insensitive to situations in which some customers are poorly served whereas others fully served; introducing a quality of service measured per single customer might improve the fairness of the resulting solutions. Finally, the insertion of dynamic and stochastic features in the problem: in the current version of the problem, dynamic occurrences such as alarm triggering or unexpected urgent services are not considered; by embedding them into a new problem that considers dynamic and stochastic aspects, we may obtain a more sophisticated model, to be used on-the-fly during the execution of the activities. The large availability of data from the company makes this last research direction very interesting.

Conclusion

In this thesis, we have focused on the development of models and methods to create decision support systems by means of a mathematical approach and heuristic algorithm to solve problems derived from real business cases in distributed logistics systems.

In Chapter 1, we have presented the study of the Time Windows Assignment Vehicle Routing Problem (TWAVRP), a VRP variant that appears when the volume of customer demands is uncertain and visits over multiple days should be planned. The objective was to create routes that minimize expected travel costs, assigning a time window over all scenarios to each customer, and respecting the vehicle capacity. We decided to begin our research with the development of a good and flexible metaheuristic, and to test it on the benchmark TWAVRP instances, to check if good-quality solutions can be found within reasonable computational efforts. To this aim, we have proposed an Iterated Local Search (ILS) algorithm that generates a pool of feasible routes for each scenario, and a mathematical model, called Route Selector Model (RSM), that chooses the most appropriate routes, among those created, in order to minimize total costs and indicate the time windows for the customers. We compared the results of our algorithm (ILS+RSM) with the Branch-and-Cut proposed by Dalmeijer and Spliet (2018). The ILS+RSM presented competitive results, concerning both solution quality and computational effort, in particular for the larger size instances involving 45 and 50 customers.

Chapter 2 showed a mathematical model to solve a scheduling problem during the Covid-19 pandemic. This chapter studied a real-life personnel scheduling problem, motivated by Covid-19 pandemic, from a large Italian pharmaceutical distribution warehouse operated by Coopservice. A MILP formulation was proposed for the problem, which was solved using an open-source optimization software, namely CBC from COIN-OR. The optimal solution obtained by our formulation was capable of considerably improving the schedule adopted by the company. Because Coopservice spends a large part of its budget on human resources (more than 25,000 employees), it is crucial to have a strategy to properly manage the personnel, especially during emergency situations such as epidemic crises. In general, this problem faces the issue of reducing the risk of contagion and organizing the schedule of the shifts in these types of scenarios.

In Chapter 3, we have shown a real-life task and personnel scheduling problem arising in Coopservice to provide cleaning services inside a hospital. We proposed an iterative three-step procedure to solve a real-life problem of integrated task and personnel scheduling aiming to minimize the total labor cost. By implementing this approach, the solution obtained by our formulation was capable of improving the schedule adopted by the company in an acceptable CPU time for this very complex problem. The results are preliminary and yet they are very positive from the company's perspective, which is now able to generate and simulate many different scenarios.

In Chapter 4, the reader was able to find a case study regarding lead time prediction in the pharmaceutical logistics sector of the company. In particular, we compared support vector machines, random forests, multi-layer perceptron, linear regression, and k-nearest neighbors on a very large collection of examples provided by a large company with headquarters in Italy. Our experimental results are very encouraging, showing how the purchasing lead time can be forecast with high accuracy, especially for linear support vector regression. In particular, the use of simple non-linear approaches does not seem to yield significant improvements in forecasting. An accurate forecast of such lead time can be crucial for decision-making, optimization, and planning in the overall pharmaceutical supply chain. Waiting times for drugs and medicines could be reduced, and hospitals and pharmacies could choose the most convenient supplier at every moment based on accurate predictions. This can be very relevant

when treating patients with urgent needs, as well as fast-changing medical conditions, such as the ones we are currently facing in the COVID-19 pandemic.

In Chapter 5, a real-life multi-period orienteering problem related to the activity of patrolling a vast area to provide security services was presented. The resulting optimization problem is a challenging variant of a Multi-Period Orienteering Problem. Due to the difficulty of the problem, we have chosen to solve it by means of an ILS equipped with an inner VND. We have tested the ILS on real-world instances provided by the company. The ILS was able to considerably reduce the kilometers traveled by the patrols increasing the number of visits to the customers. The average improvement in the objective function value was 47.3%, and not only the minimum QoS was always respected, but it was also increased on average by about 10%. These good results have been obtained by a simple but effective combination of several algorithmic components, including six different local search algorithms. By performing a sensitivity analysis on the entire set of instances, we could assess the fact that all such local searches have a positive contribution to the ILS performance.

Bibliography

Albers, Casper J, Frank Critchley, and John C Gower (2011). "Quadratic minimisation problems in statistics". In: *Journal of Multivariate Analysis* 102.3, pp. 698–713.

Aledort, Julia E, Nicole Lurie, Jeffrey Wasserman, and Samuel A Bozzette (2007). "Non-pharmaceutical public health interventions for pandemic influenza: an evaluation of the evidence base". In: *BMC public health* 7.1, p. 208.

Alonso, MT, R Alvarez-Valdes, M Iori, and F Parreño (2019). "Mathematical models for multi container loading problems with practical constraints". In: *Computers & Industrial Engineering* 127, pp. 722–733.

Amorim, Pedro, Sophie N. Parragh, Fabricio Sperandio, and Bernardo Almada-Lobo (2014). "A rich vehicle routing problem dealing with perishable food: A case study". In: *TOP* 22, pp. 489–508.

Angelelli, Enrico, Claudia Archetti, Carlo Filippi, and Michele Vindigni (2017). "The probabilistic orienteering problem". In: *Computers & Operations Research* 81, pp. 269–281.

— (2021). "A dynamic and probabilistic orienteering problem". In: *Computers & Operations Research* 136, p. 105454.

Archetti, Claudia, Francesco Carrabs, and Raffaele Cerulli (2018). "The set orienteering problem". In: *European Journal of Operational Research* 267.1, pp. 264–272.

Archetti, Claudia, M Grazia Speranza, and Daniele Vigo (2014). "Vehicle routing problems with profits". In: *in: P. Toth and D. Vigo, eds., Vehicle Routing: Problems, Methods, and Applications, second edition*. Milano: SIAM, pp. 273–297.

Avci, Mustafa and Seyda Topaloglu (2017). "A multi-start iterated local search algorithm for the generalized quadratic multiple knapsack problem". In: *Computers & Operations Research* 83, pp. 54 –65. ISSN: 0305-0548.

Averweg, Udo Richard (2010). "Decision support systems and decision-making processes". In: *Business Information Systems: Concepts, Methodologies, Tools and Applications*. IGI Global, pp. 135–143.

Azi, Nabila, Michel Gendreau, and Jean-Yves Potvin (2010). "An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles". In: *European Journal of Operational Research* 202.3, pp. 756–763.

Bartolini, Enrico, Mauro Dell'Amico, and Manuel Iori (2017). "Scheduling cleaning activities on trains by minimizing idle times". In: *Journal of Scheduling* 20.5, pp. 493–506.

Bell, David E (1982). "Regret in decision making under uncertainty". In: *Operations research* 30.5, pp. 961–981.

Belton, Valerie and Theodor Stewart (2002). *Multiple criteria decision analysis: an integrated approach*. Springer Science & Business Media.

Bergh, Jorne Van den, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, and Liesje De Boeck (2013). "Personnel scheduling: A literature review". In: *European Journal of Operational Research* 226.3, pp. 367–385.

Berling, Peter and Mojtaba Farvid (2014). "Lead-time investigation and estimation in divergent supply chains". In: *International Journal of Production Economics* 157, pp. 177–189.

Borgonovo, Emanuele (2010a). "A methodology for determining interactions in probabilistic safety assessment models by varying one parameter at a time". In: *Risk Analysis: An International Journal* 30.3, pp. 385–399.

— (2010b). "Sensitivity analysis with finite changes: An application to modified EOQ models". In: *European Journal of Operational Research* 200.1, pp. 127–138.

Bräysy, Olli and Michel Gendreau (2005a). "Vehicle routing problem with time windows, Part I: Route construction and local search algorithms". In: *Transportation Science* 39.1, pp. 104–118.

— (2005b). "Vehicle routing problem with time windows, Part II: Metaheuristics". In: *Transportation Science* 39.1, pp. 119–139.

Breiman, Leo, Jerome Friedman, Charles J Stone, and Richard A Olshen (1984). *Classification and regression trees*. CRC press.

Brown, Marie T, Jennifer Bussell, Suparna Dutta, Katherine Davis, Shelby Strong, and Suja Mathew (2016). "Medication adherence: truth and consequences". In: *The American journal of the medical sciences* 351.4, pp. 387–399.

Bruck, Bruno P, Valerio Incerti, Manuel Iori, and Matteo Vignoli (2017). "Minimizing CO2 emissions in a practical daily carpooling problem". In: *Computers & Operations Research* 81, pp. 40–50.

Brucker, Peter, Rong Qu, and Edmund Burke (2011). "Personnel scheduling: Models and complexity". In: *European Journal of Operational Research* 210.3, pp. 467 –473. ISSN: 0377-2217.

Cabrera-Sánchez, Juan-Pedro and Ángel F Villarejo-Ramos (2020). "Acceptance and use of big data techniques in services companies". In: *Journal of Retailing and Consumer Services* 52.C, p. 101888.

Caceres-Cruz, Jose, Pol Arias, Daniel Guimarans, Daniel Riera, and Angel A. Juan (Dec. 2014). "Rich Vehicle Routing Problem: Survey". In: *ACM Computing Surveys* 47.2. ISSN: 0360-0300.

Campana, P Nicolas, Giorgio Zucchi, Manuel Iori, Carlo A Magni, and Anand Subramanian (2021). "An integrated task and personnel scheduling problem to optimize distributed services in hospitals". In: *Proceedings of the 23th International Conference on Enterprise Information Systems (ICEIS 2021)* 1, pp. 461–470.

Ceschia, Sara, Luca Di Gaspero, and Andrea Schaerf (2011). "Tabu search techniques for the heterogeneous vehicle routing problem with time windows and carrier-dependent costs". In: *Journal of Scheduling* 14.6, pp. 601–615.

Chamikara, MAP, P Bertok, D Liu, S Camtepe, and I Khalil (2020). "Efficient privacy preservation of big data for accurate data mining". In: *Information Sciences* 527, pp. 420–443. ISSN: 0020-0255.

Chang, F-CR (1997). "Heuristics for dynamic job shop scheduling with real-time updated queueing time estimates". In: *International Journal of Production Research* 35.3, pp. 651–665.

Chung, Wenming, Srinivas Talluri, and Gyöngyi Kovács (2018). "Investigating the effects of lead-time uncertainties and safety stocks on logistical performance in a border-crossing JIT supply chain". In: *Computers & Industrial Engineering* 118, pp. 440–450.

Coopservice Scpa (2020). URL: https://www.coopservice.it/.

Cordeau, J-F, G Laporte, and A Mercier (2001). "A unified tabu search heuristic for vehicle routing problems with time windows". In: *Journal of the Operational Research Society* 52.8, pp. 928–936.

Cordeau, Jean-François, Michel Gendreau, and Gilbert Laporte (1997). "A tabu search heuristic for periodic and multi-depot vehicle routing problems". In: *Networks: An International Journal* 30.2, pp. 105–119.

"Coronavirus disease 2019 (COVID-19): A literature review" (2020). In: *Journal of Infection and Public Health* 13.5, pp. 667 –673.

Dalmeijer, Kevin and Remy Spliet (2018). "A branch-and-cut algorithm for the Time Window Assignment Vehicle Routing Problem". In: *Computers & Operational Research* 89, pp. 140–152.

Dean, Jared (2014). *Big data, data mining, and machine learning: value creation for business leaders and practitioners*. John Wiley & Sons.

Desrochers, Martin, Jacques Desrosiers, and Marius Solomon (1992). "A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows". In: *Operations Research* 40.2, pp. 342–354.

Doi, Tsubasa, Tatsushi Nishi, and Stefan Voß (2018). "Two-level decomposition-based matheuristic for airline crew rostering problems with fair working time". In: *European Journal of Operational Research* 267.2, pp. 428–438.

Drucker, Harris, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik (1997). "Support vector regression machines". In: *Advances in neural information processing systems 9*. Ed. by M. C. Mozer, M. I. Jordan, and T. Petsche. MIT Press, pp. 155–161.

Eberle, Lukas Gallus, Hirokazu Sugiyama, and Rainer Schmidt (2014). "Improving lead time of pharmaceutical production processes using Monte Carlo simulation". In: *Computers & Chemical Engineering* 68, pp. 255–263.

El-Rifai, Omar, Thierry Garaix, and Xiaolan Xie (2016). "Proactive on-call scheduling during a seasonal epidemic". In: *Operations Research for Health Care* 8, pp. 53–61.

Elahipanah, Mahsa, Guy Desaulniers, and Eve Lacasse-Guay (Oct. 2013). "A two-phase mathematical-programming heuristic for flexible assignment of activities and tasks to work shifts". In: *Journal of Scheduling* 16, pp. 443–460.

Ernst, Andreas, Houyuan Jiang, Mohan Krishnamoorthy, Bowie Owens, and David Sier (2004a). "An annotated bibliography of personnel scheduling and rostering". In: *Annals of Operations Research* 127.1-4, pp. 21–144.

Ernst, Andreas, Houyuan Jiang, Mohan Krishnamoorthy, and David Sier (2004b). "Staff scheduling and rostering: A review of applications, methods and models". In: *European Journal of Operational Research* 153.1, pp. 3–27.

Ernst, Andreas T, Houyuan Jiang, Mohan Krishnamoorthy, and David Sier (2004c). "Staff scheduling and rostering: A review of applications, methods and models". In: *European Journal of Operational Research* 153.1, pp. 3–27.

Fang, Yaqing, Yiting Nie, and Marshare Penny (2020). "Transmission dynamics of the COVID-19 outbreak and effectiveness of government interventions: A data-driven analysis". In: *Journal of medical virology* 92.6, pp. 645–659.

Fayyad, Usama, Gregory Piatetsky-Shapiro, and Padhraic Smyth (1996). "From data mining to knowledge discovery in databases". In: *AI magazine* 17.3, p. 37.

Fırat, Murat and Cor AJ Hurkens (2012). "An improved MIP-based approach for a multi-skill workforce scheduling problem". In: *Journal of Scheduling* 15.3, pp. 363–380.

Flahault, Antoine, Elisabeta Vergu, Laurent Coudeville, and Rebecca F Grais (2006). "Strategies for containing a global influenza pandemic". In: *Vaccine* 24.44-46, pp. 6751–6755.

Frank, Alejandro Germán, Lucas Santos Dalenogare, and Néstor Fabián Ayala (2019). "Industry 4.0 technologies: Implementation patterns in manufacturing companies". In: *International Journal of Production Economics* 210, pp. 15–26.

Gatica, Gabriel, Nilay Shah, and Lazaros G. Papageorgiou (2001). "Capacity planning under clinical trials uncertainty for the pharmaceutical industry". In: *European Symposium on Computer Aided Process Engineering - 11*. Ed. by Rafiqul Gani and Sten Bay Jørgensen. Vol. 9. Computer Aided Chemical Engineering. Elsevier, pp. 865 –870.

Gavalas, Damianos, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou (2014). "A survey on algorithmic approaches for solving tourist trip design problems". In: *Journal of Heuristics* 20.3, pp. 291–328.

Gendreau, Michel, Gilbert Laporte, and Frédéric Semet (1998). "A tabu search heuristic for the undirected selective travelling salesman problem". In: *European Journal of Operational Research* 106.2-3, pp. 539–545.

Golden, Bruce L, Larry Levy, and Rakesh Vohra (1987). "The orienteering problem". In: *Naval Research Logistics (NRL)* 34.3, pp. 307–318.

Goltsos, Thanos E, Borja Ponte, Shixuan Wang, Ying Liu, Mohamed M Naim, and Aris A Syntetos (2019). "The boomerang returns? Accounting for the impact of uncertainties on the dynamics of remanufacturing systems". In: *International Journal of Production Research* 57.23, pp. 7361–7394.

Gunawan, Aldy, Hoong Chuin Lau, and Kun Lu (2015a). "An Iterated Local Search Algorithm for Solving the Orienteering Problem with Time Windows". In: *Evolutionary Computation in Combinatorial Optimization*. Ed. by Gabriela Ochoa and Francisco Chicano. Cham: Springer International Publishing, pp. 61–73.

— (2015b). *Well-Tuned ILS for Extended Team Orienteering Problem with Time Windows*. Tech. rep. LARC-TR-01-15. Available at http://research.larc.smu.edu.sg/larcweb/larc/publications/technicalreports/Well-Tuned-ILS-for-Extended-Team-Orienteering-Problem-with-Time-WindowsTR-01-15.pdf: Living Analytics Research Center.

Gunawan, Aldy, Hoong Chuin Lau, and Pieter Vansteenwegen (2016). "Orienteering problem: A survey of recent variants, solution approaches and applications". In: *European Journal of Operational Research* 255.2, pp. 315–332. ISSN: 03772217.

Gündling, Felix and Tim Witzel (2020). "Time-Dependent Tourist Tour Planning with Adjustable Profits". In: *Proc. 20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS) Pisa, Italy, September 7-8*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Gurobi Optimization, LLC (2020). *Gurobi Optimizer Reference Manual*. URL: http://www.gurobi.com.

Guyon, O., P. Lemaire, É. Pinson, and D. Rivreau (2010). "Cut generation for an integrated employee timetabling and production scheduling problem". In: *European Journal of Operational Research* 201.2, pp. 557 –567. ISSN: 0377-2217.

Gyulai, Dávid, András Pfeiffer, Gábor Nick, Viola Gallina, Wilfried Sihn, and László Monostori (2018). "Lead time prediction in a flow-shop environment with analytical and machine learning approaches". In: *IFAC-PapersOnLine* 51.11, pp. 1029–1034.

Haddadene, Syrine Roufaida Ait, Nacima Labadie, and Caroline Prodhon (2016). "A GRASP x ILS for the vehicle routing problem with time windows, synchronization and precedence constraints". In: *Expert Systems with Applications* 66, pp. 274 –294. ISSN: 0957-4174.

Hansen, Pierre, Nenad Mladenović, Jack Brimberg, and José A Moreno Pérez (2019). "Variable neighborhood search". In: *in: M.Gendrau and J.-Y. Potvin, eds., Handbook of Metaheuristics*. New York: Springer, pp. 57–97.

Harapan, Harapan, Naoya Itoh, Amanda Yufika, Wira Winardi, Synat Keam, Heyhpeng Te, Dewi Megawati, Zinatul Hayati, Abram L Wagner, and Mudatsir Mudatsir (2020). "Coronavirus disease 2019 (COVID-19): A literature review". In: *Journal of infection and public health*.

Haugh, Kathy Henley (2014). "Medication adherence in older adults: The pillbox half full". In: *Nursing Clinics of North America* 49.2, pp. 183–199.

Hoffmann, Kirsten, Udo Buscher, Janis Sebastian Neufeld, and Felix Tamke (2017). "Solving practical railway crew scheduling problems with attendance rates". In: *Business & Information Systems Engineering* 59.3, pp. 147–159.

Hosoda, Takamichi and Stephen M Disney (2018). "A unified theory of the dynamics of closed-loop supply chains". In: *European Journal of Operational Research* 269.1, pp. 313–326.

Hu, Xingchen, Witold Pedrycz, and Xianmin Wang (2018). "Fuzzy classifiers with information granules in feature space and logic-based computing". In: *Pattern Recognition* 80, pp. 156–167.

Ioannou, George and Stavrianna Dimitriou (2012). "Lead time estimation in MRP/ERP for make-to-order manufacturing systems". In: *International Journal of Production Economics* 139.2, pp. 551–563.

Ishizaka, Alessio and Philippe Nemery (2013). *Multi-criteria decision analysis: methods and software*. John Wiley & Sons.

Juette, Silke and Ulrich W Thonemann (2012). "Divide-and-price: A decomposition algorithm for solving large railway crew scheduling problems". In: *European Journal of Operational Research* 219.2, pp. 214–223.

Jun, Hong-Bae, Jin-Young Park, and Hyo-Won Suh (2006). "Lead time estimation method for complex product development process". In: *Concurrent Engineering* 14.4, pp. 313–328.

Kabugo, James Clovis, Sirkka-Liisa Jämsä-Jounela, Robert Schiemann, and Christian Binder (2020). "Industry 4.0 based process data analytics platform: A waste-to-energy plant case study". In: *International Journal of Electrical Power & Energy Systems* 115, p. 105508.

Keskin, Merve and Bülent Çatay (2018). "A matheuristic method for the electric vehicle routing problem with time windows and fast chargers". In: *Computers & Operations Research* 100, pp. 172–188.

Khodadadian, M, A Divsalar, C Verbeeck, A Gunawan, and P Vansteenwegen (2022). "Time dependent orienteering problem with time windows and service time dependent profits". In: *Computers and Operations Research* 143.March, p. 105794. ISSN: 0305-0548.

Kim, Sun Hoon, Jeong Woo Kim, and Young Hoon Lee (2014). "Simulation-based optimal production planning model using dynamic lead time estimation". In: *The International Journal of Advanced Manufacturing Technology* 75.9-12, pp. 1381–1391.

Kletzander, Lucas and Nysret Musliu (2020). "Solving the general employee scheduling problem". In: *Computers & Operations Research* 113, p. 104794.

Kotiloglu, Serhan, Theodoros Lappas, Konstantinos Pelechrinis, and PP Repoussis (2017). "GW2020". In: *Tourism Management* 62, pp. 76–88.

Kramer, Raphael, Jean-François Cordeau, and Manuel Iori (2019). "Rich vehicle routing with auxiliary depots and anticipated deliveries: An application to pharmaceutical distribution". In: *Transportation Research Part E: Logistics and Transportation Review* 129, pp. 162–174.

Krishnamoorthy, M., A.T. Ernst, and D. Baatar (2012). "Algorithms for large scale Shift Minimisation Personnel Task Scheduling Problems". In: *European Journal of Operational Research* 219.1, pp. 34 –48. ISSN: 0377-2217.

Kuo, Chung-Feng Jeffrey, Chieh-Hung Lin, and Ming-Hao Lee (2018). "Analyze the energy consumption characteristics and affecting factors of Taiwan's convenience stores-using the big data mining approach". In: *Energy and Buildings* 168, pp. 120–136.

Lapègue, Tanguy, Odile Bellenguez-Morineau, and Damien Prot (2013). "A constraint-based approach for the shift design personnel task scheduling problem with equity". In: *Computers & Operations Research* 40.10, pp. 2450–2465. ISSN: 0305-0548.

Laporte, Gilbert and François V. Louveaux (1993). "The integer L-shaped method for stochastic integer programs with complete recourse". In: *Operations Research Letters* 13.3, pp. 133–142.

Li, Heng, Shang-Ming Liu, Xiao-Hua Yu, Shi-Lin Tang, and Chao-Ke Tang (2020). "Coronavirus disease 2019 (COVID-19): current status and future perspectives". In: *International journal of antimicrobial agents* 55.5, p. 105951.

Lingitz, Lukas, Viola Gallina, Fazel Ansari, Dávid Gyulai, András Pfeiffer, and László Monostori (2018). "Lead time prediction using machine learning algorithms: A case study by a semiconductor manufacturer". In: *Procedia CIRP* 72, pp. 1051–1056.

López-Ibáñez, Manuel, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas G Stützle (2016). "The irace package: Iterated racing for automatic algorithm configuration". In: *Operations Research Perspectives* 3, pp. 43–58.

Lougee-Heimer, Robin (2003). "The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community". In: *IBM Journal of Research and Development* 47.1, pp. 57–66.

Lourenço, Helena R, Olivier C Martin, and Thomas Stützle (2003). "Iterated local search". In: *Handbook of metaheuristics*. Springer, pp. 320–353.

Lourenço, Helena Ramalhinho, Olivier C Martin, and Thomas Stützle (2019). "Iterated local search: Framework and applications". In: *in: M.Gendrau and J.-Y. Potvin, eds., Handbook of Metaheuristics*. New York: Springer, pp. 129–168.

Lu, Weisheng, Xi Chen, Yi Peng, and Liyin Shen (2015). "Benchmarking construction waste management performance using big data". In: *Resources, Conservation and Recycling* 105.Part A, pp. 49–58.

Maenhout, Broos and Mario Vanhoucke (2018). "A perturbation matheuristic for the integrated personnel shift and task re-scheduling problem". In: *European Journal of Operational Research* 269.3, pp. 806–823.

Marchesi, Janaina F., Silvio Hamacher, and Julia L. Fleck (2020). "A stochastic programming approach to the physician staffing and scheduling problem". In: *Computers & Industrial Engineering* 142, p. 106281.

Martins, Lucas Burahem, Manuel Iori, Mayron César O Moreira, and Giorgio Zucchi (2021). "On Solving the Time Window Assignment Vehicle Routing Problem via Iterated Local Search". In: *Graphs and Combinatorial Optimization: from Theory to Applications, CTW2020 Proceedings*, pp. 223–235.

Mitchell, Stuart, Michael O'Sullivan, and Iain Dunning (2011). "PuLP: a linear programming toolkit for python". In: *The University of Auckland, Auckland, New Zealand*. URL: http://www.optimization-online.org/DB_FILE/2011/09/3178.pdf.

Montgomery, Douglas C, Elizabeth A Peck, and G Geoffrey Vining (2012). *Introduction to linear regression analysis*. Vol. 821. Wiley Series in Probability and Statistics. John Wiley & Sons.

Moreira, Mayron César O and Alysson M Costa (2013). "Hybrid heuristics for planning job rotation schedules in assembly lines with heterogeneous workers". In: *International Journal of Production Economics* 141.2, pp. 552–560.

Moscelli, Giuseppe, Luigi Siciliani, and Valentina Tonei (2016). "Do waiting times affect health outcomes? Evidence from coronary bypass". In: *Social Science & Medicine* 161, pp. 151–159.

Mourtzis, Dimitris, Michael Doukas, Katerina Fragou, Kostas Efthymiou, and Violeta Matzorou (2014). "Knowledge-based estimation of manufacturing lead time for complex engineered-to-order products". In: *Procedia CIRP* 17, pp. 499–504.

Neves-Moreira, Fábio, Diogo Pereira da Silva, Luís Guimarães, Pedro Amorim, and Bernardo Almada-Lobo (2018). "The time window assignment vehicle routing problem with product dependent deliveries". In: *Transportation Research Part E: Logistics and Transportation Review* 116, pp. 163 –183.

Nogueira, Bruno, Rian G. S. Pinheiro, and Anand Subramanian (2018). "A Hybrid Iterated Local Search heuristic for the maximum weight independent set problem". In: *Optimization Letters* 12.3, pp. 567–583.

Noori-Daryan, Mahsa, Ata Allah Taleizadeh, and Fariborz Jolai (2019). "Analyzing pricing, promised delivery lead time, supplier-selection, and ordering decisions of a multinational supply chain under uncertain environment". In: *International Journal of Production Economics* 209, pp. 236–248.

Oliveira, Maiza B, Giorgio Zucchi, Marco Lippi, Douglas F Cordeiro, Núbia R Silva, and Manuel Iori (2021). "Lead Time Forecasting with Machine Learning Techniques for a Pharmaceutical Supply Chain". In: *Proceedings of the 23th International Conference on Enterprise Information Systems (ICEIS 2021)* 1, pp. 634–641.

Ozkarahan, Irem and James E Bailey (1988). "Goal programming model subsystem of a flexible nurse scheduling support system". In: *IIE transactions* 20.3, pp. 306–316.

Öztürk, Atakan, Sinan Kayalıgil, and Nur E Özdemirel (2006). "Manufacturing lead time estimation using data mining". In: *European Journal of Operational Research* 173.2, pp. 683–700.

Palomo-Martínez, Pamela J, M Angélica Salazar-Aguilar, Gilbert Laporte, and André Langevin (2017). "A hybrid variable neighborhood search for the orienteering problem with mandatory visits and exclusionary constraints". In: *Computers & Operations Research* 78, pp. 408–419.

Pazour, Jennifer A. and Russell D. Meller (2013). "Exploring the Parallels Between a Hospital Pharmacy and a Distribution Center". In: *Systems Analysis Tools for Better Health Care Delivery*. Ed. by Panos M. Pardalos, Pando G. Georgiev, Petraq Papajorgji, and Britta Neugaard. New York, NY: Springer New York, pp. 131–150. ISBN: 978-1-4614-5094-8.

Pfeiffer, András, Dávid Gyulai, Botond Kádár, and László Monostori (2016). "Manufacturing lead time estimation with the combination of simulation and statistical learning methods". In: *Procedia CIRP* 41, pp. 75–80.

Ponte, Borja, José Costas, Julio Puche, Raúl Pino, and David de la Fuente (2018). "The value of lead time reduction and stabilization: A comparison between traditional and collaborative supply chains". In: *Transportation Research Part E: Logistics and Transportation Review* 111, pp. 165–185.

Ristoski, Petar and Heiko Paulheim (2016). "Semantic Web in data mining and knowledge discovery: A comprehensive survey". In: *Journal of Web Semantics* 36, pp. 1–22.

Rumelhart, D. E. and J. L. McClelland (1987). "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. MIT Press, pp. 318–362.

Sagaert, Yves R, Nikolaos Kourentzes, Stijn De Vuyst, El-Houssaine Aghezzaf, and Bram Desmet (2019). "Incorporating macroeconomic leading indicators in tactical capacity planning". In: *International Journal of Production Economics* 209, pp. 12–19.

Salazar-González, Juan-José (2014). "Approaches to solve the fleet-assignment, aircraft-routing, crew-pairing and crew-rostering problems of a regional carrier". In: *Omega* 43, pp. 71–82.

Samanta, Sukanya, Goutam Sen, and Soumya Kanti Ghosh (2022a). "A literature review on police patrolling problems". In: *Annals of Operations Research, 316*, pp. 1063–1106.

Samanta, Sukanya, Goutam Sen, and Soumya Kanti Ghosh (2022b). "Correction to: A literature review on police patrolling problems". In: *Annals of Operations Research, 316*, p. 1575.

Savelsbergh, Martin WP (1992). "The vehicle routing problem with time windows: Minimizing route duration". In: *ORSA Journal on Computing* 4.2, pp. 146–154.

Seccia, Ruggiero (2020). *The Nurse Rostering Problem in COVID-19 emergency scenario*. Tech. rep. DIAG - Sapienza University of Rome. URL: http://www.optimization-online.org/DB_HTML/2020/03/7712.html.

Shereen, Muhammad Adnan, Suliman Khan, Abeer Kazmi, Nadia Bashir, and Rabeea Siddique (2020). "COVID-19 infection: origin, transmission, and characteristics of human coronaviruses". In: *Journal of Advanced Research* 24, pp. 91–98. ISSN: 2090-1232.

Sievers, Stefan, Tim Seifert, Marcel Franzen, Gerhard Schembecker, and Christian Bramsiepe (2017). "Lead time estimation for modular production plants". In: *Chemical Engineering Research and Design* 128, pp. 96–106.

Sivarajah, Uthayasankar, Muhammad Mustafa Kamal, Zahir Irani, and Vishanth Weerakkody (2017). "Critical analysis of Big Data challenges and analytical methods". In: *Journal of Business Research* 70, pp. 263–286.

Smet, Pieter, Andreas T Ernst, and Greet Vanden Berghe (2016). "Heuristic decomposition approaches for an integrated task scheduling and personnel rostering problem". In: *Computers & Operations Research* 76, pp. 60–72.

Spliet, Remy, Said Dabia, and Tom Van Woensel (2018). "The Time Window Assignment Vehicle Routing Problem with Time-Dependent Travel Times". In: *Transportation Science* 52.2, pp. 261–276.

Spliet, Remy and Guy Desaulniers (2015). "The discrete time window assignment vehicle routing problem". In: *European Journal of Operational Research* 244.2, pp. 379–391.

Spliet, Remy and Adriana F. Gabor (2014). "The Time Window Assignment Vehicle Routing Problem". In: *Transportation Science* 49.4, pp. 721–731.

Strijbosch, LWG, RMJ Heuts, and MLJ Luijten (2002). "Cyclical packaging planning at a pharmaceutical company". In: *International Journal of Operations & Production Management* 22.5, pp. 549–564.

Subramanian, Anand and Lucídio dos Anjos Formiga Cabral (2008). "An ILS Based Heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Limit". In: *Evolutionary Computation in Combinatorial Optimization*. Ed. by Jano van Hemert and Carlos Cotta. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 135–146. ISBN: 978-3-540-78604-7.

Subramanian, Anand, Lúcia Maria de A Drummond, Cristiana Bentes, Luiz Satoru Ochi, and Ricardo Farias (2010). "A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery". In: *Computers & Operations Research* 37.11, pp. 1899–1911.

Tatsiopoulos, IP and BG Kingsman (1983). "Lead time management". In: *European Journal of Operational Research* 14.4, pp. 351–358.

Tetteh, Ebenezer Kwabena (2019). "Reducing avoidable medication-related harm: What will it take?" In: *Research in Social and Administrative Pharmacy* 15.7, pp. 827–840.

Tohidi, Mohammad, Masoumeh Kazemi Zanjani, and Ivan Contreras (2020). "A physician planning framework for polyclinics under uncertainty". In: *Omega*, p. 102275.

Toth, Paolo and Daniele Vigo (2014). *Vehicle routing: problems, methods, and applications*. Ed. by Paolo Toth and Daniele Vigo. 2nd. SIAM, p. 463. ISBN: 9781611973587.

Tsai, Chih-Fong, Wei-Chao Lin, and Shih-Wen Ke (2016). "Big data mining with parallel computing: A comparison of distributed and MapReduce methodologies". In: *Journal of Systems and Software* 122, pp. 83–92.

Tsiligirides, Theodore (1984). "Heuristic methods applied to orienteering". In: *Journal of the Operational Research Society* 35.9, pp. 797–809.

Van den Bergh, Jorne, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, and Liesje De Boeck (2013). "Personnel scheduling: A literature review". In: *European Journal of Operational Research* 226.3, pp. 367 –385. ISSN: 0377-2217.

Vance, Pamela H, Cynthia Barnhart, Ellis L Johnson, and George L Nemhauser (1997). "Airline crew scheduling: A new formulation and decomposition algorithm". In: *Operations Research* 45.2, pp. 188–200.

Vandaele, Nico, Liesje De Boeck, and Dominiek Callewier (2002). "An open queueing network for lead time analysis". In: *IIE transactions* 34.1, pp. 1–9.

Vanden Berghe, Greet (2002). "An advanced model and novel meta-heuristic solution methods to personnel scheduling in healthcare". PhD thesis. Leuven, Belgium: KU Leuven. URL: https://lirias.kuleuven.be/retrieve/88598.

Vansteenwegen, Pieter, Wouter Souffriau, and Dirk Van Oudheusden (2011). "The orienteering problem: A survey". In: *European Journal of Operational Research* 209.1, pp. 1–10.

Vidal, T., T. G. Crainic, M. Gendreau, and C. Prins (2013). "Heuristics for multi-attribute vehicle routing problems: A survey and synthesis". In: *European Journal of Operational Research* 231.1, pp. 1–21.

Vidal, Thibaut, Teodor Crainic, Michel Gendreau, and Christian Prins (2012). *A unifying view on timing problems and algorithms*. Tech. Rep. 43, CIRRELT.

Vidal, Thibaut, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins (2014). "A unified solution framework for multi-attribute vehicle routing problems". In: *European Journal of Operational Research* 234.3, pp. 658–673.

Volk, Rebekka, Julian Stengel, and Frank Schultmann (2014). "Building Information Modeling (BIM) for existing buildings—Literature review and future needs". In: *Automation in construction* 38, pp. 109–127.

Xu, Wenzheng, Weifa Liang, Zichuan Xu, Jian Peng, Dezhong Peng, Tang Liu, Xiaohua Jia, and Sajal K. Das (2021). "Approximation algorithms for the generalized team orienteering problem and its applications". In: *IEEE/ACM Transactions on Networking* 29.1, pp. 176–189.

Zhang, Shu, Jeffrey W Ohlmann, and Barrett W Thomas (2020). "Multi-period orienteering with uncertain adoption likelihood and waiting at customers". In: *European Journal of Operational Research* 282.1, pp. 288–303.

Zhigalov, Grigory (2018). *VRPTW*. GitHub repository: https://github.com/donfaq/VRPTW, last commit: 0261f086d1610c2da3152540d0e535a4eaeef76c.

Zucchi, G., V.H.V. Correa, A.G. Santos, M. Iori, and M. Yagiura (2022). "A Metaheuristic Algorithm for a Multi-period Orienteering Problem arising in a Car Patrolling Application". In: *Proc. 10th International Network Optimization Conference (INOC) Aachen, Germany, March 1-4*, pp. 99–104.

Zucchi, Giorgio, Manuel Iori, and Anand Subramanian (2021). "Personnel scheduling during Covid-19 pandemic". In: *Optimization Letters* 15, pp. 1385–1398.