






An Integrated Task and Personnel Scheduling Problem to Optimize Distributed Services in Hospitals

Nícolás P. Campana¹^a, Giorgio Zucchi^{2,3}^b, Manuel Iori⁴^c, Carlo Alberto Magni⁵^d
and Anand Subramanian⁶^e

¹*Departamento de Ciência da Computação, Universidade Federal de Lavras, Av. Central, Campus Universitário, 37200-000, Lavras (MG), Brazil*

²*FMB, Marco Biagi Foundation, University of Modena and Reggio Emilia, Largo Marco Biagi 10, 41121 Modena, Italy*

³*R&D Department, Coopservice S.coop.p.a, Via Rochdale 5, 42122 Reggio Emilia, Italy*

⁴*DISMI, Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, 42122 Reggio Emilia, Italy*

⁵*Department of Economics “Marco Biagi”, University of Modena and Reggio Emilia, 41121 Modena, Italy*

⁶*Departamento de Sistemas de Computação, Centro de Informática, Universidade Federal da Paraíba, Rua dos Escoteiros, Mangabeira, 58055-000, João Pessoa-PB, Brazil*

Keywords: Personnel Scheduling, Distributed Services, Mathematical Model, Metaheuristic.


Abstract: This paper addresses a real-life task and personnel scheduling problem arising in a large Italian company 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.


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 et al., 2017) that belongs to the general field of personnel scheduling (Van den 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 sanitification. 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

^a <https://orcid.org/0000-0001-5499-6876>

^b <https://orcid.org/0000-0002-5459-7290>

^c <https://orcid.org/0000-0003-2097-6572>

^d <https://orcid.org/0000-0003-3066-8426>

^e <https://orcid.org/0000-0002-9244-9969>

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.

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 1.

Step 1

Optimal allocation of cleaning tasks to days

Step 2

Creation of cleaning routes minimizing travel distance

Step 3

Personnel scheduling minimizing total labor cost

Figure 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 paper 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 paper is organized as follows. Section 2 gives a formal problem definition. Section 3 provides a brief literature review. Section 4 describes our three-step approach. Section 5 reports the results of our algorithm and, finally, Section 6 presents the concluding remarks.

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 $D = \{1, \dots, 7\}$ 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 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:

$$\min Z = h \cdot z \quad (1)$$

where z is the total working time. Let λ_e be the total working time of an employee $e \in E$. Then, minimization of (1) boils down to minimization of (2) below:

$$\min z = \sum_{e \in E} \lambda_e \quad (2)$$

The Integrated Task and Personnel Scheduling Problem (ITPSP) aims at optimize (2) while meeting the operational constraints defined at steps 1, 2 and 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 et al., 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., 2004), for the problem terminology); (ii) in the TSRP only one task can be assigned to a shift. In (Lapègue et al., 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 et al., 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 et al., 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.

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 Thone-mann, 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 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 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 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 (2) (and, therefore, minimizing the total cost).

Algorithm 1 provides an overview of the proposed approach.

Algorithm 1: Iterative three-step algorithm.

```

1: procedure 3-STEP( $U, E, \epsilon, \eta, \phi$ )
2:    $s^* \leftarrow \emptyset$  ▷ Final solution
3:    $H \leftarrow \emptyset$  ▷ Set of weekly patterns
4:   while time limit  $\epsilon$  not reached do
5:      $m \leftarrow \text{buildNewMILPModel}(H, U)$ 
6:      $\text{WeeklyPattern} \leftarrow \text{LS}(m)$ ;
7:      $C\_Schedule \leftarrow \text{VSPTW}(\text{WeeklyPattern}, \eta, \phi)$ 
8:      $P\_Schedule \leftarrow \text{PSP}(C\_Schedule, E)$ 
9:     if  $f(P\_Schedule) \leq f(s^*)$  then
10:        $s^* \leftarrow P\_Schedule$ 
11:     end if
12:      $H \leftarrow H \cup \text{WeeklyPattern}$ 
13:   end while
14:   return  $s^*$ 
15: end procedure

```

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 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).

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 et al., 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 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:

$$\min \quad z_1 \quad (3)$$

$$\sum_{p \in P_v} x_{vp} = 1 \quad \forall v \in U \quad (4)$$

$$\sum_{v \in U} \theta_{vd} Q_{pd} x_{vp} \leq z_1 \quad \forall d \in D \quad (5)$$

$$x_{vp} \in \{0, 1\} \quad \forall v \in U, \forall p \in P_v \quad (6)$$

$$z_1 \geq 0. \quad (7)$$

The objective function (3) aims at finding a balanced solution by minimizing the maximum working time across all days. Constraints (4) impose that exactly one pattern must be selected for each location v . Constraints (5) compute the maximum daily working time. Finally, constraints (6) and (7) define the domain of the variables.

At each iteration τ of Algorithm 1, the MILP model is modified by inserting a hamming-distance constraint (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)-(7)

$$\sum_{(v,p) \in s_\tau} (1 - x_{vp}) + \sum_{(v,p) \notin s_\tau} x_{vp} \geq \xi \quad s_\tau \in \mathcal{T} \quad (8)$$

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

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 2).

To solve the VSPTW, we have designed a heuristic procedure whose pseudocode is described in Algorithm 2. 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 2: VSPTW.

```

1: procedure VSPTW( $N, \eta, \phi$ )
2:    $\mathcal{T} \leftarrow \emptyset$ 
3:   for  $d \in D$  do
4:      $s \leftarrow \text{GreedyAlgorithm}(N_d)$ 
5:      $s \leftarrow \text{RVND}(s, \eta, \phi)$ 
6:      $\mathcal{T} \leftarrow \mathcal{T} \cup s$ 
7:   end for
8:   return  $\mathcal{T}$ 
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:

$$ST_k = \max(ST_{k-1} + q_{k-1}, a_k) \quad k \in \sigma \quad (9)$$

$$WT_k = WT_{k-1} + (ST_k - ST_{k-1} - q_{k-1}) \quad k \in \sigma \quad (10)$$

$$FD_k = \min(FD_{k-1}, u_k - ST_k + WT_k) \quad k \in \sigma. \quad (11)$$

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.)

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 3 describes the procedure developed to the referred subproblem.

Algorithm 3: 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 et al., 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 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 2, we present a graphical representation of the proposed algorithm in order to help the reader understand the presented methodology.

Let $D=\{1,\dots,7\}$ be the set of days. For each location v we have a set of cleaning task T_v and a set P_v of feasible patterns

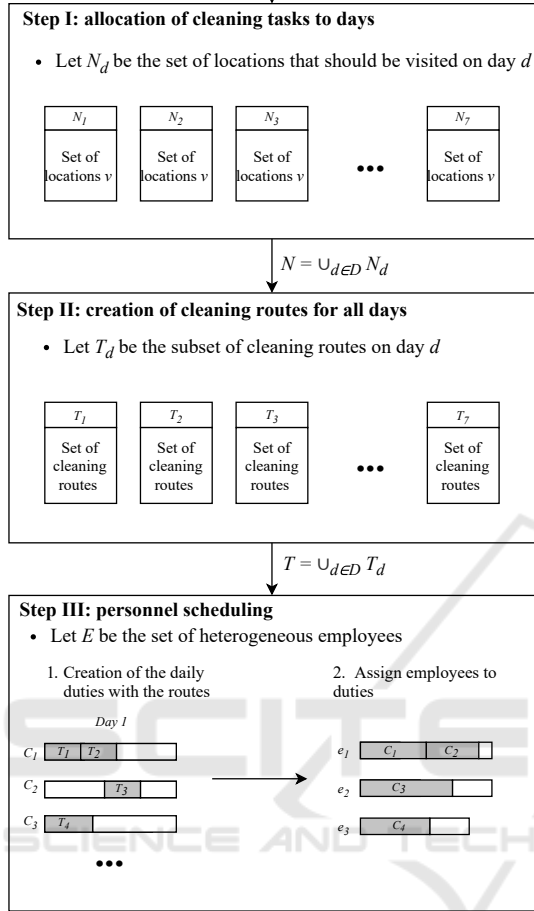


Figure 2: An overview of the proposed algorithm.

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 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 2 using 4 cores, benefiting from the fact that the

VSPTW can be addressed independently for each day.

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 $\epsilon = 1,800$ 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 5.3.

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 et al., 2014). In the hospital associated with the real-life instance (see Figure 3), 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 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

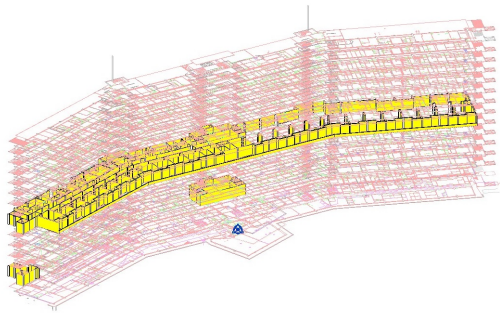


Figure 3: An overview of the hospital.

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 (12) calculates the shortest path from i to j :

$$SP(i, j) = \min_{\kappa_1, \kappa_2 \in \hat{N}^b} d_{i\kappa_1}^{\mathcal{L}(i)} + d_{\kappa_2 j}^{\mathcal{L}(j)}. \quad (12)$$

The cost c_{ij} is therefore given by:

$$c_{ij} = \begin{cases} SP(i, j) & \text{if } \mathcal{L}(i) \neq \mathcal{L}(j) \\ d_{ij} & \text{otherwise} \end{cases} \quad (13)$$

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

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 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 (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 $100(z^c - z)/z^c$.

5.4 Results for the Artificial Instances

We also carried out a similar experiment for the artificial instances. For each instance, we considered

Table 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

the same values of η and ξ specified in the previous section. Table 2 reports the information of each instances, namely, the number of floors and locations, together with the combination of η and ξ that yielded the best result. In the last two columns, we provide the value of the objective function and the CPU time required to find the best solution. The results show that, on average, 1186 seconds are necessary for the method to obtain the best solution. 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 2: Best results found on the artificial instances.

Instance	Floor	Locations	η^{best}	ξ^{best}	z	t^*
1	4	534	30	50	46,336	568.73
2	11	1,678	30	20	126,520	634.37
3	6	857	30	50	66,136	1,646.40
4	3	483	40	50	38,936	1,355.59
5	14	2,020	20	20	164,512	1,705.86
6	5	743	40	1	51,098	209.24
7	10	1,451	20	50	106,348	844.65
8	3	494	30	20	40,442	1,671.83
9	3	439	20	1	31,626	997.60
10	11	1,613	20	50	132,850	1,487.25
11	11	1,643	30	1	128,154	662.20
12	3	399	30	1	43,918	1,577.16
13	9	1,226	40	20	106,160	176.98
14	5	777	20	20	81,944	1,716.20
15	15	2,191	20	20	180,794	969.58
16	2	223	40	50	23,728	1,483.14
17	11	1,645	30	20	126,366	1,594.48
18	6	897	20	50	70,277	1,298.90
19	13	1,915	40	1	165,114	1,457.60
20	13	1,875	20	50	150,448	1,662.42

Figure 4 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.

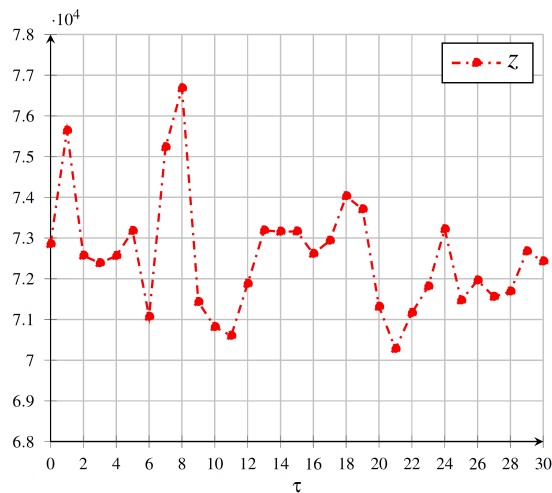


Figure 4: Variation of z for each iteration τ in the instance number 18.

6 CONCLUSIONS

In this paper, 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 et al., 2020). Future research might include: (i) the insertion of more complex cost functions in (1) that could take account of different employees' levels; (ii) 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 (suf-

ficiently large) span of time. Moreover, to improve performance in terms of solution quality, one could incorporate metaheuristic strategies, such as perturbation schemes in Algorithms 2 and 3, with a view of obtaining better local optimal solutions.

ACKNOWLEDGMENTS

We acknowledge financial support by Coopservice and by University of Modena and Reggio Emilia (grant FAR 2018). This research was also partially supported by CNPq, grants 428549/2016-0 and 307843/2018-1.

REFERENCES

- Bartolini, E., Dell'Amico, M., and Iori, M. (2017). Scheduling cleaning activities on trains by minimizing idle times. *Journal of Scheduling*, 20(5):493–506.
- Borgonovo, E. (2010a). A methodology for determining interactions in probabilistic safety assessment models by varying one parameter at a time. *Risk Analysis: An International Journal*, 30(3):385–399.
- Borgonovo, E. (2010b). Sensitivity analysis with finite changes: An application to modified eoq models. *European Journal of Operational Research*, 200(1):127–138.
- Bräysy, O. and Gendreau, M. (2005a). Vehicle routing problem with time windows, part I: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118.
- Bräysy, O. and Gendreau, M. (2005b). Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation Science*, 39(1):119–139.
- Cordeau, J.-F., Gendreau, M., and Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks: An International Journal*, 30(2):105–119.
- Doi, T., Nishi, T., and Voß, S. (2018). Two-level decomposition-based matheuristic for airline crew rostering problems with fair working time. *European Journal of Operational Research*, 267(2):428–438.
- Elahipanah, M., Desaulniers, G., and Lacasse-Guay, E. (2013). A two-phase mathematical-programming heuristic for flexible assignment of activities and tasks to work shifts. *Journal of Scheduling*, 16:443–460.
- Ernst, A. T., Jiang, H., Krishnamoorthy, M., and Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27.
- Guyon, O., Lemaire, P., Pinson, É., and Rivreau, D. (2010). Cut generation for an integrated employee timetabling and production scheduling problem. *European Journal of Operational Research*, 201(2):557–567.

- Hoffmann, K., Buscher, U., Neufeld, J. S., and Tamke, F. (2017). Solving practical railway crew scheduling problems with attendance rates. *Business & Information Systems Engineering*, 59(3):147–159.
- Juette, S. and Thonemann, U. W. (2012). Divide-and-price: A decomposition algorithm for solving large railway crew scheduling problems. *European Journal of Operational Research*, 219(2):214–223.
- Kletzander, L. and Musliu, N. (2020). Solving the general employee scheduling problem. *Computers & Operations Research*, 113:104794.
- Krishnamoorthy, M., Ernst, A., and Baatar, D. (2012). Algorithms for large scale shift minimisation personnel task scheduling problems. *European Journal of Operational Research*, 219(1):34 – 48.
- Lapègue, T., Bellenguez-Morineau, O., and Prot, D. (2013). A constraint-based approach for the shift design personnel task scheduling problem with equity. *Computers & Operations Research*, 40(10):2450–2465.
- Laporte, G. and Louveaux, F. V. (1993). The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142.
- Lougee-Heimer, R. (2003). The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66.
- Maenhout, B. and Vanhoucke, M. (2018). A perturbation matheuristic for the integrated personnel shift and task re-scheduling problem. *European Journal of Operational Research*, 269(3):806–823.
- Salazar-González, J.-J. (2014). Approaches to solve the fleet-assignment, aircraft-routing, crew-pairing and crew-rostering problems of a regional carrier. *Omega*, 43:71–82.
- Savelsbergh, M. W. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, 4(2):146–154.
- Smet, P., Ernst, A. T., and Berghe, G. V. (2016). Heuristic decomposition approaches for an integrated task scheduling and personnel rostering problem. *Computers & Operations Research*, 76:60–72.
- Subramanian, A., Drummond, L. M. d. A., Bentes, C., Ochi, L. S., and Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899–1911.
- Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., and De Boeck, L. (2013). Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367–385.
- Vance, P. H., Barnhart, C., Johnson, E. L., and Nemhauser, G. L. (1997). Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, 45(2):188–200.
- Vidal, T., Crainic, T., Gendreau, M., and Prins, C. (2012). *A unifying view on timing problems and algorithms*. Tech. Rep. 43, CIRRELT.
- Volk, R., Stengel, J., and Schultmann, F. (2014). Building information modeling (bim) for existing buildings—literature review and future needs. *Automation in construction*, 38:109–127.
- Zucchi, G., Iori, M., and Subramanian, A. (2020). Personnel scheduling during covid-19 pandemic. *Optimization Letters*, pages 1–12 (forthcoming).