Received: 29 November 2022 | Revised: 21 August 2023 | Accepted: 24 August 2023

DOI: 10.1002/net.22187

SPECIAL ISSUE ARTICLE

WILEY

An iterated local search for a multi-period orienteering problem arising in a car patrolling application

Victor Hugo Vidigal Corrêa[1,2] | **Hang Dong**[3] | **Manuel Iori**[2] | **André Gustavo dos Santos**[1] | **Mutsunori Yagiura**[3] | **Giorgio Zucchi**[4,5]

[1]Department of Informatics, Federal University of Viçosa, Viçosa, Brazil
[2]Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Reggio Emilia, Italy
[3]Graduate School of Informatics, Nagoya University, Nagoya, Japan
[4]School of Doctorate E4E, University of Modena and Reggio Emilia, Modena, Italy
[5]R&D Department, Coopservice s.c.p.a, Reggio Emilia, Italy

Correspondence
Victor Hugo Vidigal Corrêa, Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Reggio Emilia, Italy.
Email: victor.vidigal@ufv.br

Funding information
Coopservice scpa; Fundação de Amparo à Pesquisa do Estado de Minas Gerais, Grant/Award Numbers: 20H02388, 22H00513.

Abstract

This paper 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.

KEYWORDS

car patrolling problem, iterated local search, mixed integer linear programming, orienteering problem, variable neighborhood descent

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 1 shows the current customers of the Emilia Romagna region, divided by province.

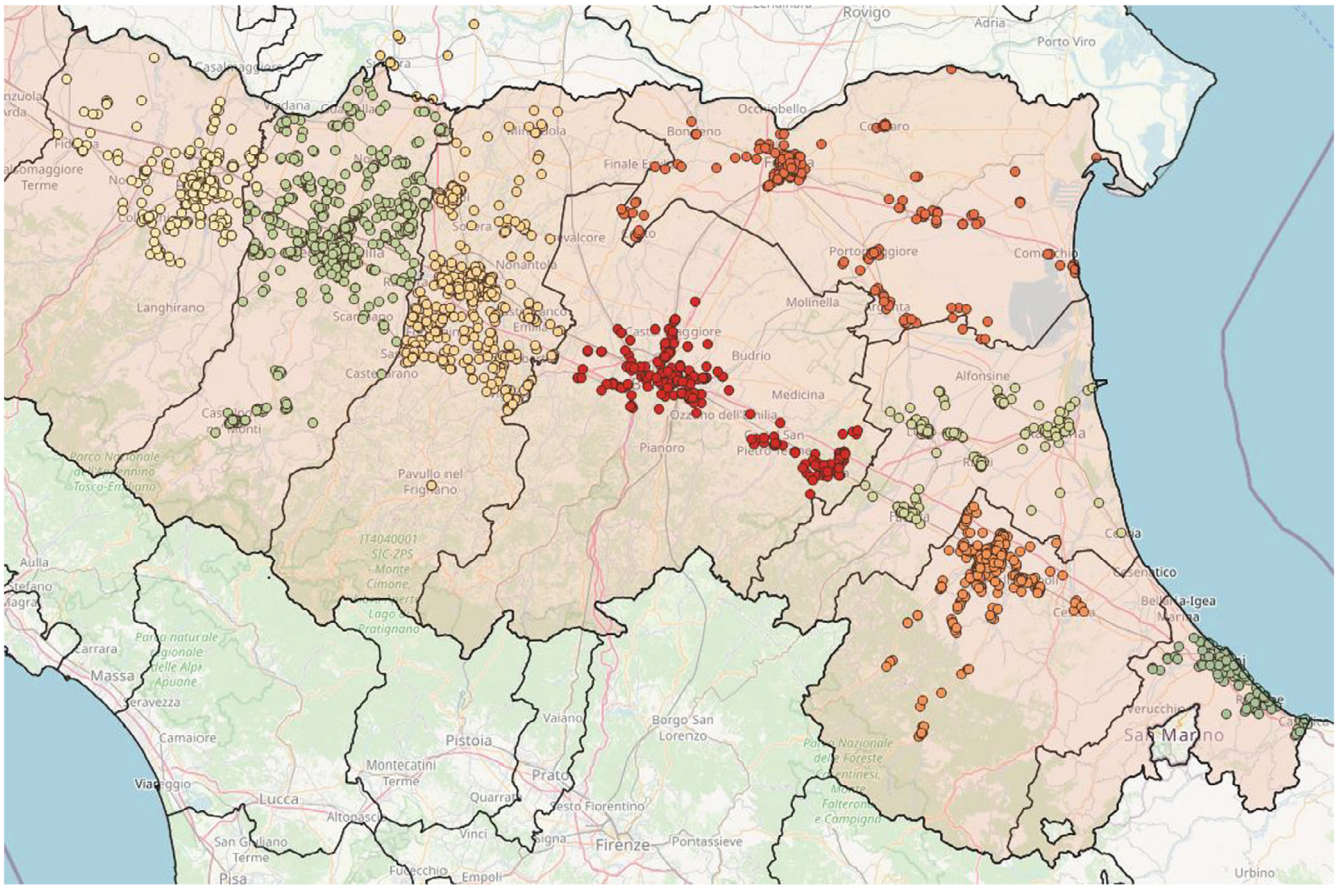

FIGURE 1 Customers in the Emilia Romagna region, divided by province.

The customers are geographically dispersed in the area and are consequently divided into clusters. Each cluster is assigned to a patrol, which performs every working shift (e.g., 8 h) a route to visit customers and execute the required services. Figure 2 provides better details for the province of Reggio Emilia, showing the customers divided into patrolling clusters. The cluster configuration does not change from a working shift 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, which corresponds to a working shift of a patrol. 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 min 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) [7]. 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 [14]. The ILS receives in input the set of customers and the set of services

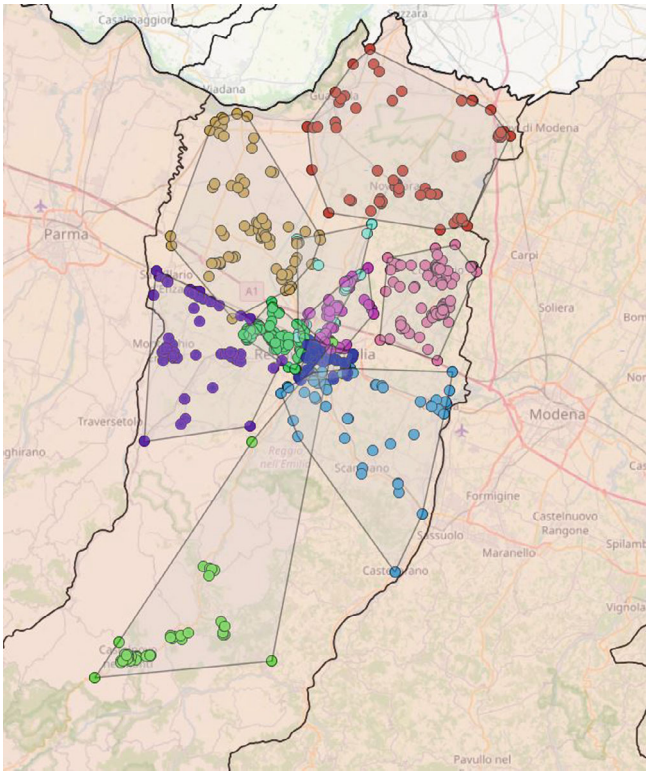

FIGURE 2 Customers in the Reggio Emilia province, divided by clusters.

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 [11] 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 2 contains a brief literature review of patrolling applications and OPs. Section 3 formally describes the problem. Section 4 presents the mathematical formulation. Section 5 gives the details of the ILS algorithm. Section 6 shows the computational results and, finally, Section 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 [23].

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, [16,17] 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 [18], 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, [6] 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 [9] and [19]. More in detail, [19] formally described the most relevant problem variants, and surveyed known exact and heuristics algorithms, whereas [9] computationally evaluated eight algorithms to solve the team orienteering problem with time windows, finding out that the ILS by [8] was the algorithm producing the best solutions on average. Surveys on related classes of problems were presented by [5], who focused on tourist trip design problems and by [4], 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 [15] 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 [1], 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 [13] 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 [3]. 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 [10], 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 [22]. 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 [2], 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 [20], 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 [12]. 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.

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 $G_0 = (C_0, A_0)$. The set of vertices is defined as $C_0 = \{0, 1, \dots, n\}$, where 0 is the depot at which the single vehicle starts and ends each route, and $C = \{1, \dots, n\}$ is the set of customers. The graph is complete and a traveling time γ'_{ij} is associated with each arc $(i, j) \in A_0$, with $\gamma'_{ii} = 0$ for each $i \in C_0$.

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 $\overline{n}_{cdt}$ be the number of services that have been actually performed in a solution. We define the quality of service (QoS) level as $Q = \sum_{c\in C,\ d\in D_c,\ t\in T_{cd}} \overline{n}_{cdt} / \sum_{c\in C,\ d\in D_c,\ t\in T_{cd}} n_{cdt}$. 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, which corresponds to the patrol working shift). For some services, such as the closing or the opening of a commercial activity, the time window is strict (e.g., 10 min) 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 min 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 period. If multiple visits are performed in the same period for the same service at the same customer, then additional scores are collected. The additional scores are computed according to the decreasing function defined next. In case a service is repeated in a different period, then the score to be collected starts again from w_{ct} for the first visit and then decreases for subsequent visits. In detail, let $\tau = 1, \dots, n_{cdt}$ 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 $w_{ct\tau} > w_{ct,\tau+1}$ for $\tau = 1, \dots, n_{cdt} - 1$. 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 $z = \alpha\mathcal{S} - \beta\mathcal{T}$.

4 | MATHEMATICAL MODEL

To model the MPOPTW as an MILP, we work on an extended graph $G = (V, A)$ 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 $d \in D$. The overall set of vertices in G is then defined as $V = \{0\} \cup \{\cup_{d\in D} V_d\} \cup \{n+1\}$.

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 of service t and the time window of customer c. The graph is complete and we associate with each arc $(i,j) \in A$ a traveling time γ_{ij} that is equal to the traveling time between the two customers associated with vertices i and j, respectively. Namely, by considering two customers $i \in V_{cdt}$ and $j \in V_{\hat{c}\hat{d}\hat{t}}$, we set $\gamma_{ij} = \gamma'_{c\hat{c}}$ (and note that, consequently, $\gamma_{ij} = 0$ when $c = \hat{c}$).

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 $d \in D$, 0 otherwise; (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:

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

$$\text{s. t.} \quad \sum_{j \in V_d^{n+1}} x_{0jd} = \sum_{i \in V_d^0} x_{i,n+1,d} = y_{0d} = y_{n+1,d} = 1 \quad d \in D \tag{2}$$

$$\sum_{i \in V_d^0} x_{ivd} = \sum_{j \in V_d^{n+1}} x_{vjd} = y_{vd} \quad v \in V_d, d \in D \tag{3}$$

$$\sum_{v \in V_{cdt}} y_{vd} = n_{cdt} \quad c \in C, d \in D_c, t \in M_{cd} \tag{4}$$

$$s_{id} + q_i + \gamma_{ij} - \mathcal{M}(1 - x_{ijd}) \leq s_{jd} \quad i \in V_d^0, j \in V_d^{n+1}, d \in D \tag{5}$$

$$e_{id} - \mathcal{M}(1 - y_{id}) \leq s_{id} \quad i \in V_d^0, d \in D \tag{6}$$

$$s_{jd} \leq l_{jd} + \mathcal{M}(1 - y_{jd}) \quad j \in V_d^{n+1}, d \in D \tag{7}$$

$$s_{jd} - s_{id} \geq \delta_{\min} - \mathcal{M}(2 - y_{id} - y_{jd}) \quad i,j \in V_{cdt} : i < j, t \in T_{cd}, c \in C, d \in D \tag{8}$$

$$\frac{1}{n} \sum_{d \in D} \sum_{v \in V_d} y_{vd} \geq Q_{\min} \tag{9}$$

$$x_{ijd} \in \{0,1\} \quad i \in V_d^0, j \in V_d^{n+1}, d \in D \tag{10}$$

$$y_{vd} \in \{0,1\} \quad v \in V_d \cup \{0, n+1\}, d \in D \tag{11}$$

$$s_{id} \geq 0 \quad i \in V_d \cup \{0, n+1\}, d \in D. \tag{12}$$

The objective function (1) maximizes the weighted sum of total collected score minus total working time, multiplied by, respectively, α and β. Constraints (2) guarantee that each route starts and finishes at the depot. Constraints (3) guarantee route connectivity and also enforce the relation between variables x and y. Constraints (4) guarantee that all mandatory services are executed. Constraints (5), (6), and (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 (8) guarantee that visits for the same service required by a customer are separated by at least $\delta_{\min}$ units of time. Constraint (9) imposes the minimum QoS. Constraints (10), (11), and (12) give the domain of the decision variables.

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

ALGORITHM 1. ILS algorithm

```
1:  procedure ILS(T_max = max run time, I_max = max number of iterations without improvements, p = perturbation intensity)
2:      ITERATION ← 0                                   ▷ the number of iterations without improvement
3:      s* ← CONSTRUCTIVEHEURISTIC
4:      s* ← VND(s*)
5:      while ELAPSEDTIME ≤ T_max and ITERATION ≤ I_max do
6:          s' ← PERTURBATION(s*, p)
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
```

invokes local search procedures with a set of neighborhoods and finds a locally optimal solution for this set [11]. 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 1. 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.

Evaluation function. Our ILS uses the objective function as it is to evaluate solutions. Let $S(\sigma_d)$ 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:

$$z(s) = \alpha \sum_{d \in D} S(\sigma_d) - \beta \sum_{d \in D} \mathcal{T}(\sigma_d). \tag{13}$$

Constructive heuristic. An initial solution is constructed by a greedy algorithm, whose pseudo-code is summarized in Algorithm 2. 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.

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 3 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.

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;

ALGORITHM 2. 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
```

ALGORITHM 3. 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
```

- 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 first neighborhood, called Relocate inter-period, randomly selects an optional service that is currently performed in d_1 and could also be performed in d_2, and then it tries to relocate it to d_2. Namely, it first selects a vertex i_1 in σ_{d_1} that is associated with an optional service in both U_{c,d_1} and U_{c,d_2} for a certain $c \in C$, it removes it from σ_{d_1}, and then it 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 second neighborhood, called Swap inter-period, randomly chooses a vertex i_1 in σ_{d_1} that is associated

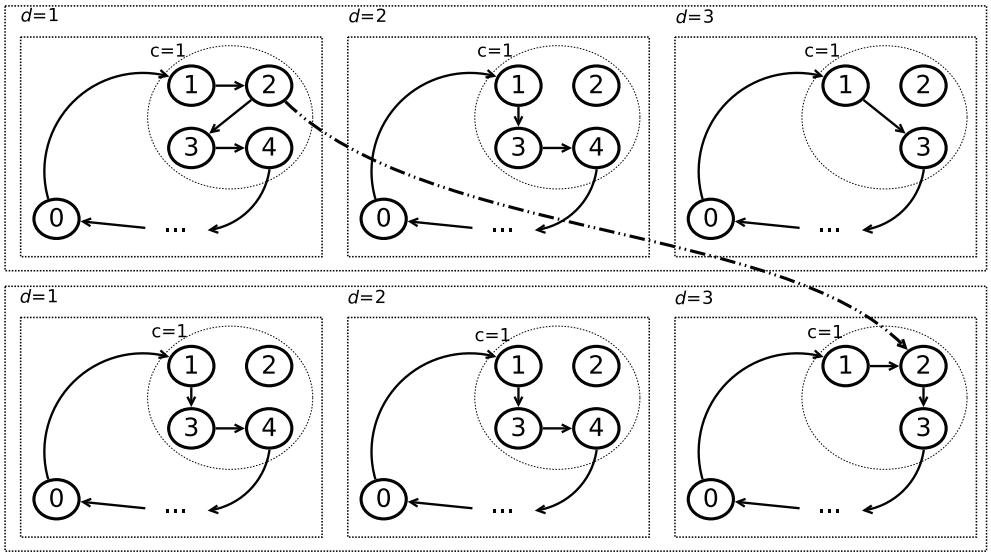

FIGURE 3 Depicting example for the relocate inter-period perturbation movement.

with an optional service in both U_{c,d_1} and U_{c,d_2} for a certain $c \in C$, and then it tries to swap it with another vertex i_2 in σ_{d_2} that is associated with an optional service in both U_{c',d_1} and U_{c',d_2} for a certain $c' \in C$. 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 a certain amount of successful moves p have been produced, or $|\sigma_{d_1}||\sigma_{d_2}|$ attempts (either successful or unsuccessful) have been performed, where $|\sigma_d|$ gives the number of vertices in σ_d.

Figure 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.

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 executed on a single thread. We are aware that Python is slower if compared to other languages such as C/C++ (see, e.g., [21]), but the company that supported this research has asked for a Python implementation because this is the language they adopt in their Research and Development department. The algorithm that we have developed is indeed in use at this department for strategic decisions. The MILP model was solved with Gurobi 9.5 and 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.

For what concerns the termination conditions, the ILS was allowed to run for at most $T_{\max} = 3600$ seconds on each instance. On the basis of preliminary computational experiments, the value of parameter $I_{\max}$ has been set to 100 (refer to Section 6.5 below for a full analysis on this parameter). The value of p has been set to 10, which led to slightly better results than 5 and 15.

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 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.

TABLE 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

TABLE 2 Average computational results obtained by the ILS on full set of 79 instances.

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
Overall average		40.4	360.4	8600.7	2002.4	1568.3	157.6	119.8	93.1	1077.5	1372.7

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 $\delta_{\min} = 90$ min. The traveling times have been obtained by computing the real-world distances using the Open Source Routing Machine application.

6.2 | ILS results

Table 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 (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 over the full set of 79 instances (which can be obtained from the table by computing the weighted average of the values in each column, considering as weight the number of instances in the corresponding line).

The ILS was able to find a feasible solution for every instance in an average time of about 23 min. The average time was around 3 min or less for Mantova, Roma and Sassari, provinces requiring a small number of services, and larger than 50 min 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

TABLE 3 Comparison between Gurobi and ILS (on subset F of 37 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
Overall average			4147.9	5248.3	22%	3502.9	4374.7	6.2%	505.9

incumbent has been found, but this time is not excessive. On average, the ILS performed around 180 iterations per run, with a maximum of 553 iterations on a small instance of the Ferrara province. 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 min, 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.

6.3 | Comparison with the mathematical model

Table 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. We computed impr. as follows: we first evaluated the improvement over each of the $|F|$ instances in each province by computing $100(z - LB)/LB$; then, we obtained the resulting impr. value per province by simply evaluating the average of the improvements. The same process was followed to obtain the overall averages in the last line, which are average values with respect to all 37 instances.

From the table, we can notice that Gurobi cannot find feasible solutions for more than half of our instances. This can be imputed to constraint (9), which makes the model non-decomposable into single periods, and to constraints (5)–(8), which all contain "big M" values. We attempted using lazy-constraints callbacks and searched for better configurations with the automated parameter configuration, but none of these attempts could improve the solver performance. More in detail, 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 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 ten provinces and an identical value for the remaining instance (Modena). The average QoS produced by Gurobi is 92.7% and that of the ILS is

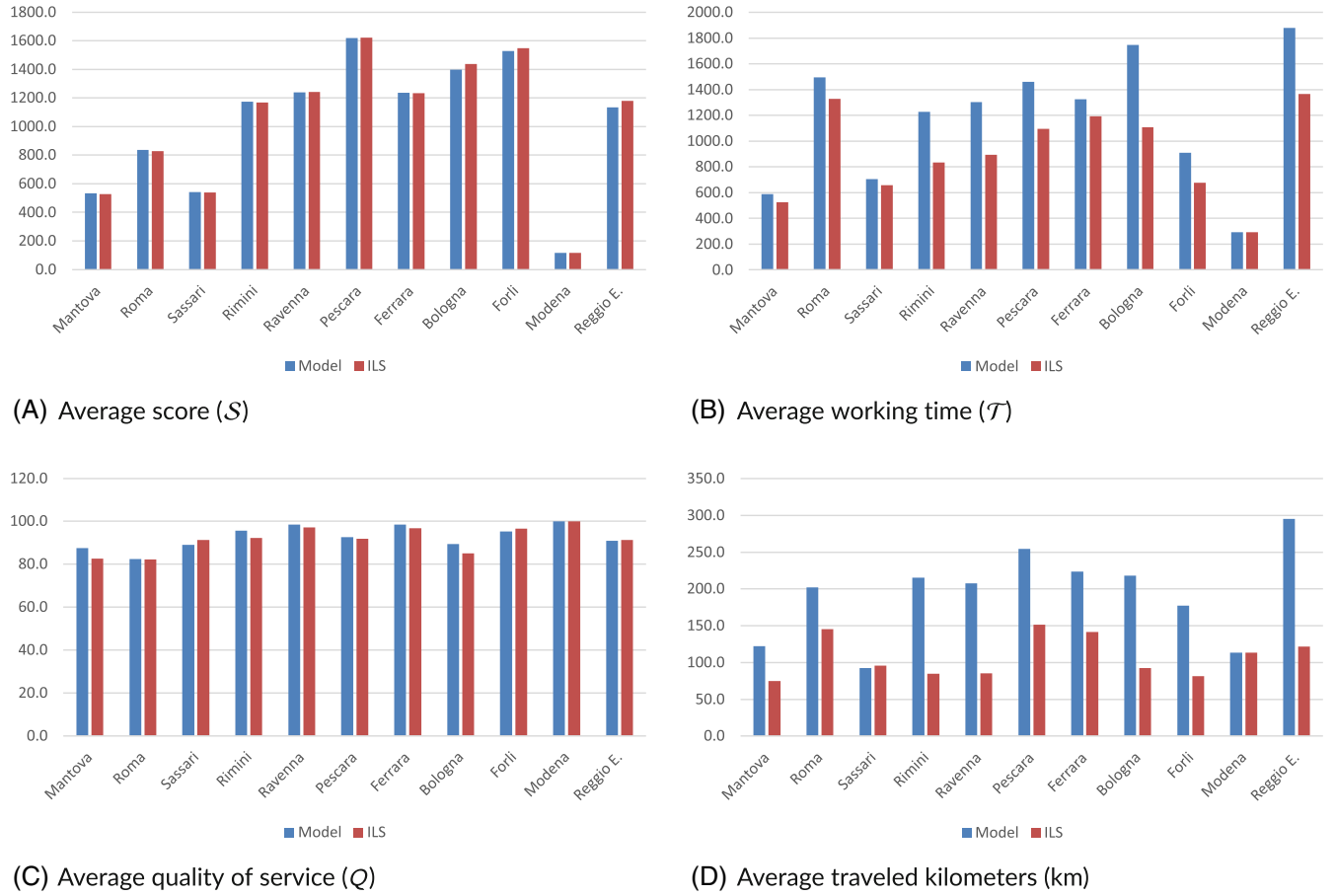

FIGURE 4 Four KPIs by Gurobi and ILS (on subset F of instances for which MILP found a feasible solution). (A) Average score (S). (B) Average working time (T). (C) Average quality of service (Q). (D) Average traveled kilometers (km).

just 91.6%. 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).

6.4 | Comparison with the company solutions

In Table 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 min, whereas the minimum required is 90 min. 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 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, Rimini, and Modena.

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, which

TABLE 4 Comparison between company and ILS solutions on the full set of 79 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	89.1	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%
Overall average	6.6	5181.6	84.8	58.4	8600.7	93.1	119.8	116%

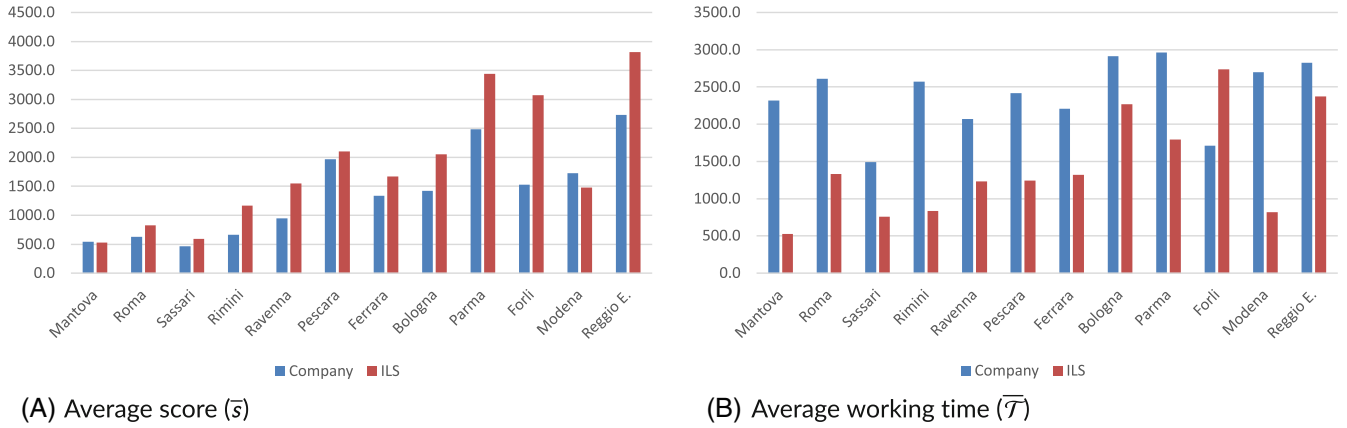

FIGURE 5 Relevant KPIs for company and ILS (on the entire set of instances). (A) Average score ($\overline{s}$). (B) Average working time ($\overline{\mathcal{T}}$).

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 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 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.

TABLE 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
Overall 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%	—

TABLE 6 Average KPIs obtained by ILS with different values of I_{max}.

I_{max}	z	$\mathcal{S}$	$\mathcal{T}$	km	δ	Q	Time$_{inc}$	Time
50	8530.6	1988.9	1570.8	157.7	114.3	93.4	880.7	1099.9
100	8600.7	2002.4	1568.3	157.6	119.8	93.1	1077.5	1372.7
150	8606.8	2002.8	1563.7	157.3	113.6	93.5	1266.1	1605.6

A key element for a good ILS performance is the acceptance criterion (refer to line 8 of Algorithm 1). In our implementation, we use the simplest criterion in which we accept a solution only if this is better than the previous one. This simple approach turned out to be slightly better than an alternative one based on the simulated annealing concept, and it was thus preferred for its ease of implementation.

Parameter I_{max} (maximum number of iterations without improvements) is another key element for our ILS. This parameter has been introduced in our implementation to limit the computation time required by the algorithm (line 5 of Algorithm 1). We set it to 100 on the basis of a set of computational experiments that we report in Table 6. The table shows the average values that we obtained on the full set of instances by attempting I_{max}=50, 100, and 150. It can be noticed that 100 provides a good improvement with respect to 50, at the expense of a limited increase in the computational effort. Setting I_{max} to 150 increases even further the time, but it does not lead to significant improvements.

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 ranged

from 23% to almost 400%, and not only the minimum QoS was always respected, but it was also increased on average from 84.8% to 93.1%.

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.

ACKNOWLEDGMENTS

A special thank to Nicolas Porto Campana and to Matteo Magnavacchi for their support in this project. This work was supported by Fundação de Amparo à Pesquisa de Minas Gerais (FAPEMIG), Coopservice Scpa, and the Japan Society for the Promotion of Science [Grants 20H02388, 22H00513]. Their support is gratefully acknowledged.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from Coopservice scpa. Restrictions apply to the availability of these data, which were used under license for this study. Data are available from the author(s) with the permission of Coopservice scpa.

ORCID

Victor Hugo Vidigal Corrêa https://orcid.org/0000-0002-8390-7075
Hang Dong https://orcid.org/0009-0002-9186-8798
Manuel Iori https://orcid.org/0000-0003-2097-6572
André Gustavo dos Santos https://orcid.org/0000-0002-5743-3523
Mutsunori Yagiura https://orcid.org/0000-0003-0970-9414
Giorgio Zucchi https://orcid.org/0000-0002-5459-7290

REFERENCES

[1] E. Angelelli, C. Archetti, C. Filippi, and M. Vindigni, *The probabilistic orienteering problem*, Comput. Oper. Res. **81** (2017), 269–281.
[2] E. Angelelli, C. Archetti, C. Filippi, and M. Vindigni, *A dynamic and probabilistic orienteering problem*, Comput. Oper. Res. **136** (2021), 105454.
[3] C. Archetti, F. Carrabs, and R. Cerulli, *The set orienteering problem*, Eur. J. Oper. Res. **267** (2018), 264–272.
[4] C. Archetti, M. G. Speranza, and D. Vigo, "*Vehicle routing problems with profits*," *Vehicle routing: Problems, methods, and applications*, second ed., P. Toth and D. Vigo (eds.), SIAM, Milano, 2014, pp. 273–297.
[5] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, *A survey on algorithmic approaches for solving tourist trip design problems*, J. Heuristics **20** (2014), 291–328.
[6] M. Gendreau, G. Laporte, and F. Semet, *A tabu search heuristic for the undirected selective travelling salesman problem*, Eur. J. Oper. Res. **106** (1998), 539–545.
[7] B. L. Golden, L. Levy, and R. Vohra, *The orienteering problem*, Naval Res. Logist. (NRL) **34** (1987), 307–318.
[8] A. Gunawan, H. C. Lau, and K. Lu, *Well-tuned ILS for extended team orienteering problem with time windows*, Tech. report LARC-TR-01-15, Living Analytics Research Center, http://research.larc.smu.edu.sg/larcweb/larc/publications/technicalreports/Well-Tuned-ILS-for-Extended-Team-Orienteering-Problem-with-Time-WindowsTR-01-15.pdf, Singapore, 2015.
[9] A. Gunawan, H. C. Lau, and P. Vansteenwegen, *Orienteering problem: A survey of recent variants, solution approaches and applications*, Eur. J. Oper. Res. **255** (2016), 315–332.
[10] F. Gündling and T. Witzel. Time-dependent tourist tour planning with adjustable profits, Proc. 20th symposium on algorithmic approaches for transportation modelling, optimization, and systems (ATMOS) Pisa, Italy. 2020.
[11] P. Hansen, N. Mladenović, J. Brimberg, and J. A. M. Pérez, "*Variable neighborhood search*," *Handbook of metaheuristics*, M. Gendrau and J.-Y. Potvin (eds.), Springer, New York, 2019, pp. 57–97.

[12] M. Khodadadian, A. Divsalar, C. Verbeeck, A. Gunawan, and P. Vansteenwegen, *Time dependent orienteering problem with time windows and service time dependent profits*, Comput. Oper. Res. **143** (2022), 105794.
[13] S. Kotiloglu, T. Lappas, K. Pelechrinis, and P. Repoussis, Gw2020*, tourism*, Manage **62** (2017), 76–88.
[14] H. R. Lourenço, O. C. Martin, and T. Stützle, "*Iterated local search: Framework and applications*," *Handbook of metaheuristics*, M. Gendrau and J.-Y. Potvin (eds.), Springer, New York, 2019, pp. 129–168.
[15] P. J. Palomo-Mart, M. A. Salazar-Aguilar, G. Laporte, and A. Langevin, *A hybrid variable neighborhood search for the orienteering problem with mandatory visits and exclusionary constraints*, Comput. Oper. Res. **78** (2017), 408–419.
[16] S. Samanta, G. Sen, and S. K. Ghosh, *A literature review on police patrolling problems*, Ann. Oper. Res. **316** (2022), 1063–1106.
[17] S. Samanta, G. Sen, and S. K. Ghosh, *Correction to: A literature review on police patrolling problems*, Ann. Oper. Res. **316** (2022), 1575.
[18] T. Tsiligirides, *Heuristic methods applied to orienteering*, J. Oper. Res. Soc. **35** (1984), 797–809.
[19] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, *The orienteering problem: A survey*, Eur. J. Oper. Res. **209** (2011), 1–10.
[20] W. Xu, W. Liang, Z. Xu, J. Peng, D. Peng, T. Liu, X. Jia, and S. K. Das, *Approximation algorithms for the generalized team orienteering problem and its applications*, IEEE/ACM Trans. Netw **29** (2021), 176–189.
[21] F. Zehra, M. Javed, D. Khan, and M. Pasha. Comparative analysis of C++ and python in terms of memory and time, Tech. report 2020120516. https://www.preprints.org/manuscript/202012.0516/v1 2020.
[22] S. Zhang, J. W. Ohlmann, and B. W. Thomas, *Multi-period orienteering with uncertain adoption likelihood and waiting at customers*, Eur. J. Oper. Res. **282** (2020), 288–303.
[23] G. Zucchi, V. Correa, A. Santos, M. Iori, and M. Yagiura. A metaheuristic algorithm for a multi-period orienteering problem arising in a car patrolling application, Proc. 10th international network optimization conference (INOC) Aachen, Germany. 2022 99–104.

How to cite this article: V. H. Vidigal Corrêa, H. Dong, M. Iori, A. G. dos Santos, M. Yagiura, and G. Zucchi, *An iterated local search for a multi-period orienteering problem arising in a car patrolling application*, Networks.. (2023), 1–16. https://doi.org/10.1002/net.22187