Optimizing a Car Patrolling Application by Iterated Local Search

Victor Hugo Vidigal Corrêa
Federal University of Viçosa and University of Modena and Reggio Emilia
Viçosa-MG, Brazil
victor.vidigal@ufv.br

Thiago Alves de Queiroz
Federal University of Catalão
Catalão-GO, Brazil
taq@ufcat.edu.br

Manuel Iori
University of Modena and Reggio Emilia
Reggio Emilia, Italy
manuel.iori@unimore.it

André Gustavo dos Santos
Federal University of Viçosa
Viçosa-MG, Brazil
andresantos@ufv.br

Mutsunory Yagiura
University of Nagoya
Nagoya, Japan
yagiura@nagoya-u.jp

Giorgio Zucchi
Coopservice S.C.p.A.
Reggio Emilia, Italy
giorgio.zucchi@coopservice.it

ABSTRACT

We address a car patrolling application arising in a service company that needs to visit customers in a large area periodically. Customers are divided into clusters, each of which is assigned to a single patrol and requires different services, either mandatory or optional, on a weekly basis. The services have to be performed by satisfying several operational constraints, including interdependent time windows and maximum route duration. The aim is to maximize a weighted function that combines the profit collected from the optional services and the total working time required to perform all routes. The resulting optimization problem can be represented as a territory design and multi-period team orienteering problem. We solve the problem using an Iterated Local Search that invokes several inner procedures, including dedicated heuristics for creating the initial clusters, perturbation operators to diversify the search, and a variable neighborhood descent to search for good-quality routes. Extensive computational tests on a set of real-world instances involving up to a few hundred customers and a few thousand services prove the algorithm efficiency.

CCS CONCEPTS

• **Applied computing** → **Transportation**; • **Mathematics of computing** → *Combinatorial optimization.*

KEYWORDS

Vehicle Routing, Car Patrolling, Territory Design, Iterated Local Search

ACM Reference Format:
Victor Hugo Vidigal Corrêa, Thiago Alves de Queiroz, Manuel Iori, André Gustavo dos Santos, Mutsunory Yagiura, and Giorgio Zucchi. 2024. Optimizing a Car Patrolling Application by Iterated Local Search. In *Genetic and Evolutionary Computation Conference (GECCO '24), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3638529.3654080

GECCO '24, July 14–18, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0494-9/24/07.
https://doi.org/10.1145/3638529.3654080

1 INTRODUCTION

With the lack of public security and the growing violence, the private security market has been constantly increasing. It is estimated that half of the planet lives in countries where private security workers are more in number than public ones. By 2017, this industry was worth $ 180 billion, and it is estimated that it will keep growing in the following years [7].

In Italy, Coopservice (https://www.coopservice.it/), a large service provider, is inserted in this market with its patrolling services provided to a set of customers spread among a number of Italian provinces. Counting on more than 20 000 employees, the company provides not only security services, but also a wide range of other services, such as logistics, cleaning, and maintenance.

Regarding car patrolling, the company operates a number of security patrols that are shipped to inspect structures, parks, buildings, and many other facilities so as to counter potential criminal actions or simply restore normal, safe conditions after breakdowns. These services, such as closing and opening a commercial activity or checking the condition of a private house, are requested every week by the customers who contracted the company for their security. Some of these services are mandatory, while others are optional and, when performed, induce a score. The company's goal is to maximize the total collected score and minimize the total working time of the patrols while meeting several operational constraints, such as interdependent time windows, minimum quality of service, and maximum route duration. The quality of service is measured by the number of services performed over all clusters divided by the overall number of services requested, and it must not drop below a given threshold value.

The customers who require the services are geographically dispersed in a large area and consequently divided into clusters. Each cluster is assigned to a unique patrol, which performs one route every day to visit customers and execute the requested services.

Figure 1 details the customers' distribution in the province of Parma (Italy), showing how customers are divided into clusters. The cluster configuration does not change from one day to another, but the routes performed inside the clusters may change according to the daily service demand. Indeed, customers may require different services according to the day of the week, following the contract stipulated by the company. In a previous research on this problem [31], the cluster configuration adopted by the company

was considered as a fixed input, and the resulting routing problem was solved for each cluster independently from the other clusters, using an Iterated Local Search (ILS). In this work, we extend that previous research by proposing changes in the clusters and hence integrating the multi-period routing component of the problem with that of cluster design (also known as territory design), thus seeking solutions on the entire set of clusters. We thus expect solutions of higher quality but to be obtained at the expense of a more complex problem.

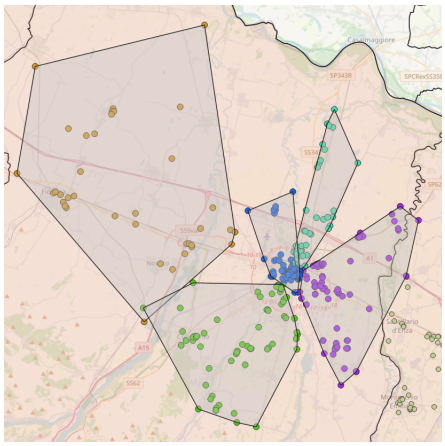

Figure 1: Customers in the Parma province, divided by clusters (solution adopted by the company).

The resulting optimization problem involves a number of operational constraints. Each customer may require multiple services and possibly multiple visits during the same period for each such service. The services must be performed within hard time windows, and each route performed by a patrol should not exceed a maximum working time. In case a customer requires multiple visits for the same service in the same period, two consecutive visits should be separated by at least a given threshold time (e.g., 90 minutes or so).

During the execution of their activities, patrols may have to perform services that were not initially scheduled, like the verification of the triggering of an alarm. This could also be considered part of the optimization, but it would complicate the problem even more since it would introduce a dynamic stochastic aspect. In our work, we disregard alarms and leave them as a future research avenue. Instead, we focus on developing an algorithm that can be used strategically to define a good cluster configuration for customers in a region. Good territory partitioning, defined for a medium-long term, is the basis for obtaining high-quality daily routes for the patrols.

The first goal of our research is to develop algorithms that can produce new cluster configurations that will lead to a better provision of the patrolling services provided by the company. This can be interpreted as a clustering problem, a class of problems widely studied in the literature and tackled by several optimization algorithms. For example, clustering algorithms based on centroids, like the k-means [1], or based on density, like the Density-Based Spatial Clustering of Applications with Noise [28], are popular algorithms for this class.

Our second goal is to build a set of routes, one per cluster and per period, by optimizing a weighted objective function that takes into account both the total collected score (to be maximized) and the total working time (to be minimized). The resulting subproblem can be interpreted as a multi-period team orienteering problem, which is a generalization of the well-known and strongly NP-hard orienteering problem [9].

We solve the resulting problem by means of an ILS [17]. Within the ILS, we invoke several inner algorithms to produce tentative cluster configurations and improve the patrol routes. The routing part is largely based on a Variable Neighborhood Descent (VND) approach [13], which looks for good-quality routes by exploring larger and larger neighborhoods. The resulting algorithm has been tested on ten instances derived from the real-world activity of the service company, consistently improving the solutions currently in use.

The remainder of this paper is organized as follows. In Section 2, we delve into the literature on the problem and its two subproblems we deal with, namely clustering and routing. In Section 3, we formalize the overall problem. Then, in Section 4, we describe the solution methods we developed, and in Section 5, we present the outcome of computational experiments we performed on real-world instances encountered in different Italian provinces. Final conclusions and future research directions are outlined in Section 6.

2 LITERATURE REVIEW

Car patrolling is a security activity largely adopted to ensure that businesses can operate regularly without the worry of criminal actions overnight. Such activity can involve several operating models, and different methodologies have been developed to optimize it. In our study case, the customers require a series of security-related services to be performed by patrols. The customers are partitioned into clusters, and each patrol serves a single cluster. This gives raise to a territory design subproblem (to define the clusters served by the patrols) and a car patrolling subproblem (to define the routes for each cluster and for each period). The literature on this field is large. We cite the recent surveys in Samanta et al. [25, 26], which reviewed police patrolling problems and divided them into two main categories: route design and district design. Our application combines these two categories, resulting in a very complex problem. In the following sections, we separately explore each category, so as to enlighten the current state of the art and relate it to our problem.

2.1 Car patrolling problems

For each cluster and each period, we handle a routing subproblem that shares similarities with the Orienteering Problem (OP), a well-known generalization of the Traveling Salesman Problem (TSP) in which it is not imposed to visit all customers. The literature on OPs is large, with numerous applications. The first studies on OPs date back to Tsiligirides [29]. In the following years, heuristic methods have been largely developed due to the NP-hard nature of the OP. First, we refer to the extensive surveys in Vansteenwegen et al. [30] and Gunawan et al. [11]. In particular, Vansteenwegen et al. [30] described the most relevant OP variants and pointed out the key features of exact and heuristic methods used to solve them. A

few years later, Gunawan et al. [11] focused on the computational evaluation of eight methods to solve the team orienteering problem (TOP, the OP variant that considers multiple vehicles) with time windows. The authors concluded that the best results, on average, are obtained by the ILS in Gunawan et al. [10]. Other surveys concerning tourist trip design problems and vehicle routing problems were presented by, respectively, Gavalas et al. [8] and Archetti et al. [2].

Further literature contributions related to the problem we tackle in this work can be found in Palomo-Martínez et al. [19], Kotiloglu et al. [15], and Gündling and Witzel [12]. In Palomo-Martínez et al. [19], a hybrid heuristic composed of a greedy randomized adaptive search procedure (GRASP) and a variable neighborhood search (VNS) was proposed to solve an OP generalization. This variant imposes constraints on mandatory visits and incompatibilities among nodes. The hybrid heuristic takes advantage of the multi-start feature of the GRASP to generate initial solutions that were then optimized with the VNS.

In Kotiloglu et al. [15], the authors handled the personalized multi-period tour recommendation problem. Their aim was to generate tours containing mandatory and optional visits while maximizing the total collected score of the optional visits. The problem also considered complicating aspects such as multiple periods of visits, time windows, maximum budget, and maximum tour length. The authors presented a Mixed Integer Linear Programming (MILP) model and an iterated tabu search, both evaluated on two data sets, one taken from the literature and the other composed of real-world data.

A tourist routing problem assuming visit redundancy avoidance and time window constraints was solved instead in Gündling and Witzel [12]. The authors proposed a MILP model and an ILS metaheuristic. The ILS obtained results close to those of the MILP model (solved with Gurobi) for small-size instances, while, for larger ones, it provided better solutions in most cases.

Recently, Zhang et al. [35] worked on a multi-period OP. A two-stage heuristic solved the problem. The subset of customers to be visited was decided in the first stage, whereas next, in the second stage, a vehicle routing problem was solved considering this subset. The authors took into consideration the relationship between the customers and the salesmen since the salesmen have a series of decisions to make upon arriving at a customer site.

In Xu et al. [34], a TOP was solved by including a set of features from Internet-of-Things applications: a limited budget was imposed on the vehicles; node costs were part of the objective function in addition to edge costs; and nodes could be serviced by multiple vehicles. An OP variant, including time-dependent service profits and time-dependent travel times, was studied by Khodadadian et al. [14]. The authors proposed a MILP model and a VNS heuristic with three specialized neighborhood structures, which were validated over benchmark instances. The VNS was also applied to a case study based in the city of Shiraz (Iran).

2.2 Territory design problems

Efficient territory design plays a crucial role in addressing routing problems, especially in the management of multiple vehicles. Defining working areas for each vehicle may enhance productivity and allow drivers to become familiar with their operating areas. Besides being an important question in service provision, optimizing patrol routes according to their operating areas helps in performing the service efficiently, thus improving customer satisfaction [33]. However, this approach may reduce route flexibility, mainly when integrating time-window constraints [27], so it must be tackled with care.

The literature on territory design (also referenced as districting/zone design, see, e.g., Ríos-Mercado [22]) is not limited to routing problems. Still, it includes, e.g., school redistricting [3] and sales force design [18]. Regarding routing problems, Ríos-Mercado and Salazar-Acosta [23] clustered a geospatial zone to minimize routing costs, optimizing the territory by using routes as a guiding parameter. On the other hand, Villalba and Rotta [32] clustered territories to determine routes by solving a vehicle routing problem with time windows and multiple vehicles. Both studies are examples of how clustering geospatial data is crucial for developing effective solution methods tailored to specific (real-world) situations.

One approach for solving territory design problems in the context of routing problems is the sequential one, which first addresses the territory design problem and then solves a vehicle routing problem. Although practical and easy to implement, sequential-based approaches generally result in not-so-good solutions compared to integrated approaches that face the complete problem (i.e., as a single). At the same time, implementing integrated approaches may raise practical issues related to, e.g., solution representation, number of variables, and size of the search space.

Within this context, Ríos-Mercado and Salazar-Acosta [23] proposed a GRASP for a real-world application in a beverage distribution company. The authors designed the territory in order to minimize the scattering of clients while imposing a limit on the routing cost. The problem was solved with a sequential approach: first, territories were designed in a greedy randomized way, and next, routes were determined by solving a TSP for each territory. The solution was further optimized with a local search-based procedure. Very recently, Carlsson and Delage [5] handled the territory design problem considering uncertain customers' locations. Customers were defined according to some probabilistic functions, and the design of territories relied on historical data. Routes were solely used to evaluate the quality of the territories, aiming at balancing the route size for each territory. The authors incorporated the unknown demand distribution to estimate the vehicle workload, with the aim of estimating services such as alarm verification, where occurrences might be uncertain.

In Rodrigues and Ferreira [24], the authors solved the integrated problem, i.e., they addressed the territory design combined with the determination of vehicle routes, in the context of solid waste collection. The authors solved the territory design problem with an algorithm inspired by Coulomb's law of electromagnetism. They proposed and solved a MILP model for the routing counterpart, aiming to minimize the traveling costs. Multiple key performance indicators were used to assess the quality of the solution. A computational study demonstrated that the proposed method effectively solved the problem, making it suitable for real-life scenarios and decision-makers.

A MILP model for solving the integrated problem occurring at a parcel company was proposed in Litvinchev et al. [16]. The authors

considered specific constraints, such as time windows and pickup and delivery points. The aim of their model was to obtain a balanced territory design in such a way that the average route lengths were close to their mean. The model had issues in solving instances with a relatively high number of customers, pointing out the necessity of using metaheuristic solution methods.

In this line, Zhou et al. [36] proposed a genetic algorithm to handle a real-world application in the dairy industry. The objective was to cluster customers in regions and then determine vehicles routes of vehicles for picking and delivering dairy products.

3 PROBLEM DESCRIPTION

In this section, we formally describe the optimization problem we face, which we call *the territory-design and multi-period team orienteering problem with time windows* (TDMPTOPTW). We adopt the notation in Vidigal Corrêa et al. [31] but extend it to the territory design aspect. We are given a directed graph $G_0 = (C_0, A_0)$, in which the nodes are defined by $C_0 = \{0\} \cup C$, with 0 being the depot from where all vehicles depart and return, and $C = \{1, \ldots, |C|\}$ being the customer set. The graph is complete, and we associate with each arc $(i, j) \in A_0$ a traveling time equal to γ_{ij}, imposing $\gamma_{ii} = 0$ for each i.

We denote by T the overall set of services the company provides (e.g., closing or opening a commercial activity, checking a private house). Each service $t \in T$ is associated with a standard service time q_t, which gives the time spent by a patrol for executing such a service. The services are divided into mandatory, set M, and optional, set U, thus resulting in $T = M \cup U$.

The activities are executed along a time span of D periods, where each period $d \in D$ represents the working shift of a patrol. Each customer $c \in C$ requires services for just a subset $D_c \subseteq D$ of periods. Formally, we define $T_{cd} \subseteq T$ the subset of services to be performed at customer c on period d, and we partition it as $T_{cd} = M_{cd} \cup U_{cd}$, with $M_{cd} \subseteq M$ being made of mandatory services and $U_{cd} \subseteq U$ of optional ones. Consequently, there is not a fixed number of visits required per period, but this number may change from one period to another for each customer. Formally, we let n_{cdt} denote the number of times service t is required by customer c on period d, and we let $\bar{n}_{cdt}$ denote the number of services that have been performed in a given solution (with $0 \leq \bar{n}_{cdt} \leq n_{cdt}$).

The customers have to be partitioned into K clusters, each assigned to a unique patrol that performs the required services in the given time span, with K being an input parameter of the problem representing both the number of clusters and the fleet size.

A key component that guides the search for good TDMPTOPTW solutions is the quality of service (QoS). The QoS is defined by the ratio of services that have been performed in all periods and clusters, namely, $Q = \sum_{c \in C,\, d \in D_c,\, t \in T_{cd}} \bar{n}_{cdt} / \sum_{c \in C,\, d \in D_c,\, t \in T_{cd}} n_{cdt}$. To impose that a minimum QoS is attained, the value of Q is restricted to be greater than or equal to a given input threshold value $Q_{\min}$.

Services in each cluster are performed when the patrol arrives at a customer. More in detail, a service t required by customer c in period d is associated with a given time window $[e_{cdt}, l_{cdt}]$, which represents the earliest and latest possible times to start the execution of each of the n_{cdt} services. Arriving at the customer after l_{cdt} is not allowed. Arriving before e_{cdt} is, instead, allowed, but the patrol will have to wait until e_{cdt}.

A time window $[e_0, l_0]$ is associated with the depot and is used to impose the maximum working time of each period, which also corresponds to the maximum route duration. Customers may require multiple visits for the same nodes during the same period (e.g., to check their property twice or more per night). In those cases, the start times of any two such visits should be separated by at least a given threshold $\delta_{\min}$. The resulting time windows for the successive visits are thus interdependent [6], and the threshold value serves to obtain a balanced patrol so that the visits are not too close to each other.

For each cluster and each period, a patrol starts its route at the depot, visits some (or all) customers to execute mandatory and some (or all) optional services, and eventually returns to the depot. The working time of a route is defined as the difference between the time the vehicle returns to the depot and the beginning of the shift (i.e., e_0).

A weight profit function is adopted to decide which service to select if not all of them can be conveniently processed. In detail, each service t required by customer 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. However, these are computed using a decreasing function. Namely, by letting $\tau = 1, \ldots, n_{cdt}$ be the index of the τth visit performed, for $c \in C$, $t \in T$ and $d \in D$, the score collected at visit τ is denoted by $w_{ct\tau}$ and is computed as $w_{ct1} = w_{ct}$ and $w_{ct\tau} = w_{ct} e^{1-\tau}$. This is imposed to obtain a balanced number of visits among customers and services.

Overall, the TDMPTOPTW asks to partition the customers into K clusters and to build a set of routes, one for each cluster and each period, to perform all mandatory services and some (or all) optional services within their interdependent time windows. The aim is to maximize a weighted objective function that considers the score $\mathcal{S}$ of the services that have been actually performed on all clusters and all periods minus the total working time $\mathcal{T}$ used by the entire set of routes. Formally, the objective function is defined as

$$\max z = \alpha \mathcal{S} - \beta \mathcal{T}, \tag{1}$$

with α and β being two non-negative input parameters.

4 SOLUTION ALGORITHM

To solve the TDMPTOPTW, we first take care of the territory design part by building an initial set of clusters using different methodologies (Section 4.1) and then construct the routes for all clusters by an ILS (Section 4.2). While constructing the routes, we also attempt modifying the initial clusters to explore diverse solutions.

4.1 Territory design

We consider four different methodologies to define the initial set of clusters. The first two are well-known algorithms from the clustering field, namely the constrained k-means [4] and the k-medoids [21]. Both algorithms are variants of the widely used k-means clustering algorithm. Still, they differ because the former explicitly adds constraints to the problem by imposing a minimum number

of points (i.e., customers) in each cluster. In contrast, the latter uses a medoid instead of a mean to define the clusters. In summary, the constrained k-means builds a solution by using the centroids of the clusters while minimizing the sum of the squared distances within each cluster and satisfying cardinality constraints. In contrast, the k-medoids achieves the same objective by resorting to the medoids. In our implementation, we used the γ_{ij} traveling times as a measure of the distances between points for both the constrained k-means and k-medoids.

The other two methods are based on MILP models derived from the literature on Facility Location Problems, specifically, the capacitated p-median and the capacitated p-center problems. The core idea in both methods is to associate cluster centers with facilities and then use a MILP model to decide where to locate facilities and how to assign customers to them. Both problems are solved in their capacitated version to obtain balanced clusters. We note that these models do not solve the complete problem (i.e., the TDMPTOPTW) but only the territory design part of it.

Capacitated p-median: Let $p = K$ be the number of clusters to be created and Ω a maximum number of customers per cluster, which in our implementation we set to $\Omega = \lceil |C|/p \rceil$. Let x_{ij} be a binary decision variable taking the value 1 if customer j is assigned to cluster i and 0 otherwise. Let y_i be another binary variable taking the value 1 if customer i is used to initialize a cluster and 0 otherwise. The capacitated p-median can be modeled as follows:

$$\min \quad \sum_{i \in C} \sum_{j \in C} \gamma_{ij} x_{ij} \tag{2}$$

$$\text{s. t.} \quad \sum_{i \in C} x_{ij} = 1 \quad \forall j \in C \tag{3}$$

$$\sum_{i \in C} y_i = p \tag{4}$$

$$\sum_{j \in C} x_{ij} \leq \Omega y_i \quad \forall i \in C \tag{5}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in C \tag{6}$$

$$y_i \in \{0, 1\} \quad \forall i \in C \tag{7}$$

The objective function (2) minimizes the total distance from each cluster center to each customer assigned to that cluster. Constraints (3) ensure that each customer is assigned to only one cluster. Constraints (4) ensure that exactly p clusters are opened. Constraints (5) impose a maximum number of customers to be assigned to each cluster and also impose that a customer can only be assigned to an opened cluster. Constraints (6) and (7) enforce the domain of the decision variables.

Capacitated p-center: The p-center is similar to the p-median, but it uses an additional continuous variable r that represents the maximum distance from each customer to the center of its cluster. The aim is again to assign all customers to $p = K$ clusters, but now by minimizing the maximum distance r. The resulting MILP model is as follows:

$$\min \quad r \tag{8}$$

$$\text{s. t.} \quad r \geq \sum_{i \in C} \gamma_{ij} x_{ij} \quad \forall j \in C \tag{9}$$

$$\sum_{i \in C} x_{ij} = 1 \quad \forall j \in C \tag{10}$$

$$\sum_{i \in C} y_i = p \tag{11}$$

$$\sum_{j \in C} x_{ij} \leq \Omega y_i \quad \forall i \in C \tag{12}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in C \tag{13}$$

$$y_i \in \{0, 1\} \quad \forall i \in C \tag{14}$$

$$r \geq 0 \tag{15}$$

The objective function (8) minimizes the maximum distance. Constraints (10)– (14) are equivalent to constraints (3)–(7) above, respectively. Constraints (9) and (15) state that r should be non-negative and not smaller than all selected distances.

4.2 Multi-period orienteering by ILS

Once the clusters' initial configuration has been obtained, we use an ILS algorithm to tackle the routing component of the TDMPTOPTW. Our ILS is derived from the one presented in [31], which produces a solution for a single cluster. Instead, we now consider the joint optimization of all clusters, allowing customers to move from one cluster to another during the search, thus enlarging the solutions space. The overall ILS procedure is presented in Algorithm 1. It builds an initial solution with a constructive heuristic and then iteratively applies a sequence of perturbations and local searches to find a local optimal solution. The aim of the perturbation step is to ensure that the algorithm can escape from local optima and explore different regions of the search space. Instead, the local search step aims to produce a locally optimal solution. Both procedures guarantee that the newly produced solutions are feasible; hence, no repair procedure is needed. The local search employs a VND, an algorithm that sequentially invokes local search procedures with a set of neighborhoods and finds a locally optimal solution for this set [13]. The ILS runs until a maximum run time or a maximum number of iterations without improvement is reached.

Constructive heuristic. The constructive heuristic builds an initial solution to the problem by employing a greedy mechanism. For each cluster, it sorts the services required in each period in a non-decreasing order based on the start time of their time window. Then, it builds a route for a given cluster and a given period by starting from the depot and then sequentially adding the services according to the generated order. The mandatory services are added first, then the optional ones are added, provided their inclusion is profitable and preserves the route's feasibility. Once no more service can be added to the route, the vehicle is sent back to the depot, and the heuristic proceeds by initializing a new route (for the next period, and so on for the other clusters).

Variable Neighborhood Descent. The local search consists of a VND procedure over the following sequence of neighborhood

Algorithm 1 ILS algorithm

```
1: procedure ILS(T_max = max run time, ITER = max n. of iterations, p =
   perturbation intensity, G = n. of perturbation structures)
2:     iteration ← 0  ▷ the number of iterations without improvement
3:     g ← 1
4:     s* ← ConstructiveHeuristic
5:     s* ← VND(s*)
6:     while elapsedTime ≤ T_max AND iteration ≤ ITER do
7:         s' ← Perturbation(s*, p, g)
8:         s'' ← VND(s')
9:         if Accept(s*, s'') then
10:            s* ← s''
11:            iteration ← 0
12:            g ← 1
13:        else
14:            iteration ← iteration + 1
15:            if g ≤ G then
16:                g ← g + 1
17:            end if
18:        end if
19:    end while
20:    return s*
21: end procedure
```

structures. We implemented six intra-route neighborhoods. Some are classical neighborhoods from the vehicle routing literature, whereas others are designed to meet our problem requirements. In the *remove optional* neighborhood, we remove an optional service vertex from a route, thus trying to decrease the working time. In the *swap* neighborhood, we swap the positions of two services in the same route. In the *2-opt* neighborhood, we swap two arcs in a route, reversing the visiting order of the services performed between the two arcs. In the *Relocate* neighborhood, we move a service to another position in the same route. In the *insert optional* neighborhood, we insert an optional and still unvisited service into a route to increase the collected score. In the *Swap optional* neighborhood, we swap two optional vertices by letting an unvisited vertex replace a visited one. The VND starts by exploring the first neighborhood. If no improvement is found, the search continues with the second one, and so on. If an improvement is found, the search restarts from the first neighborhood. The VND ends when no further improvement can be obtained with the last neighborhood. After preliminary experiments, we observed that all neighborhoods contributed to improving the solution values on average. At the same time, we also observed that the last two neighborhoods, namely insert optional and swap optional, required too much computing time for large instances. To obtain the best performance, we consequently imposed the last two neighborhoods to be invoked only for instances having not more than 5000 nodes.

Acceptance. An acceptance function is used at each iteration to decide whether to start the new search from the current solution or to move to a newly generated one. In our case, a newly generated solution is accepted only if its value, computed according to Equation (1), is better than that of the current solution. We also tested another common acceptance function based on a simulated annealing approach, in which worse solutions can be accepted throughout iterations according to a given probability that depends on their quality. Preliminary experiments showed that this acceptance function led to worse results than those obtained with the previous one, and we thus disregarded it.

Perturbation procedure. The perturbation procedure is a crucial component that enables the algorithm to escape from locally optimal solutions and explore other regions of the search space. We use three large inter-period and inter-cluster neighborhoods at each iteration, namely relocate inter-period, swap inter-period, and cluster change, invoked one after the other as depicted in Algorithm 2.

Algorithm 2 Perturbation procedure

```
1: procedure Perturbation(s, p, g)
2:     if g == 1 then
3:         s ← RelocateInterPeriod(s, p)
4:     else if g == 2 then
5:         s ← SwapInterPeriod(s, p)
6:     else
7:         s ← ClusterChange(s)
8:     end if
9:     return s
10: end procedure
```

In the *relocate inter-period* neighborhood, we randomly select a cluster and two periods, d_1 and d_2. We then randomly select an optional service currently performed in d_1, and that could also be performed in d_2. We try to relocate it to d_2. Namely, we first select a service i_1 in the route of period d_1 that is associated with an optional service that could be performed both in d_1 and d_2. We then remove the service from the route in d_1 and try to insert it in every position of the route in d_2.

In the *swap inter-period* neighborhood, we operate in a similar way. We randomly select a service i_1 performed in the route of period d_1 that is associated with an optional service in both periods d_1 and d_2. We then try to swap it with another service i_2 performed in day d_2. In either case, a move is applied, independently of the cost, if it succeeds in producing a feasible solution, otherwise it is rejected. Either neighborhood proceeds until a certain amount p of successful moves have been produced, or $\mu(d_1)\mu(d_2)$ attempts (either successful or unsuccessful) have been performed, where $\mu(d)$ gives the number of services performed in period d.

The *cluster change* neighborhood works differently. We create a list of all customers whose distance from the medoid of their current cluster is higher than the distance to the medoid of another cluster. All customers in the list are then moved, one at a time, from their current cluster to their closest one. Next, all routes of the clusters involved in the move are reconstructed. Unlike the two previously presented perturbations, the cluster change does not use a limited number of attempts but continues as long as there are customers to be moved.

5 COMPUTATIONAL RESULTS

In this section, we present the outcome of the computational tests we have performed to assess the performance of our solution algorithm. The algorithm was coded in Python 3.7.3 and executed

on a single thread. The MILP models were solved with Gurobi 9.5, invoked with its default configuration. The machine used for the experiments has a processor AMD Opteron(tm) 6376 (16M Cache, 2.3 GHz, 32 cores) and 64 GB of RAM. The k-means and k-medoids algorithms were implemented using Python's scikit learn package [20]. The time limit of the ILS was set to K CPU hours, with K being the given input number of clusters of the instance to be solved, while *ITER* was set to 100 after preliminary calibration tests.

The algorithm was tested on real-world instances derived from Coopservice's everyday activities. The traveling times were obtained by computing the shortest paths (between each pair of customers, and each customer and the depot) on the real road network via OpenStreetMap (https://www.openstreetmap.org/). The value of $Q_{\min}$ was set to 75%, and that of $\delta_{\min}$ to 90 minutes. The company operates a set of $|T|$ services, whose score for the first visit was set to $w_{ct} \in \{1, 6, 9, 10\}$, respectively.

We first report in Table 1 the objective function values of the solutions obtained by the constructive heuristic (i.e., step 3 of Algorithm 1) when the different clustering methods of Section 4.1 are adopted. The solution values are computed using Equation (1). Column *company* reports the values obtained by the heuristic when using the clusters currently in use at the company. In contrast, the other four columns report the values obtained with the clusters originated by the four territory design methods we implemented.

Each instance corresponds to a province in Table 1. For each such province, we show the number of clusters (#), which also corresponds to the number of patrols, the total number of customers ($|C|$), and the total number of requested services (n). The last line reports the average values of each column. The best value obtained for each instance is highlighted in bold. The ten instances provide an interesting and diversified test bed, with a number of clusters ranging from 2 to 10, a number of customers ranging from 43 to 679, and a number of requested services, either mandatory or optional, from 171 to 7812. The car patrolling activities occur over a time span of $D = 7$ days in all instances.

We observe that the clusters adopted by the company in Table 1 are a good starting point and allow the constructive heuristic to find the best solution values for two out of ten instances. The best results, on average, are obtained with the clusters originated by the constrained k-means algorithm (c. k-means for short in the table), which also leads to three out of ten best solutions. The k-medoids approach leads to less predictable results, with some low-quality solutions (e.g., for Bologna and Reggio Emilia) and some very high-quality ones (e.g., for Mantova, Pescara, Rimini, and Rome). Clusters obtained with the MILP models for the capacitated p-median and capacitated p-center are competitive only in a few instances (e.g., Mantova).

Table 2 reports the objective function values of the solutions obtained with the ILS. We have again attempted the five territory-design configurations of Table 1, but now, for each configuration, we ran the ILS with and without the cluster change (CC) perturbation procedure. The last two lines give, respectively, the average values of each column and the percentage improvement with respect to the constructive heuristic executed with the same configuration. The improvement is computed as $100(z_{ILS} - z_{constructive})/z_{constructive}$.

The results in Table 2 show that, by considering both CC and no CC, our algorithm can improve the current solutions obtained with the clusters adopted by the company (company and no CC) for eight out of ten provinces. The clusters adopted by the company prove once more to be a good starting point for the ILS, but the constrained k-means clusters provide slightly better results. Indeed, by considering the overall average solution value, we can notice that the constrained k-means outperforms all other methods. By comparing the company clusters with those generated by the constrained k-means with CC, we can also notice that the latter algorithm obtains an average improvement of 1.38% concerning the former.

The k-medoids, the p-center, and the p-median clusters can produce the best solution for Mantova, which is the smallest instance of the entire test set. The p-center also obtains the best solution value for Rome. All methods consistently improve the initial constructive heuristic, proving the efficiency of the ILS's iterative phase. The most notable improvement is obtained with the ILS running with k-medoids and CC, improving the constructive by 24%. With the exception of the p-median method, the use of CC always leads to equal or better results than those obtained without CC.

In Figure 2, we compare the computational effort required by the ILS under the constrained k-means configuration to produce the incumbent solution. For each instance, we contrast the values obtained with (in orange) and without (in blue) CC. The computational times are expressed in seconds. With the exception of Mantova, where the computational time is close to zero as the incumbent solution is found very early, the ILS needs a considerable amount of time to reach the incumbent. This is compatible with its strategic use, as its main purpose is to properly partition the customers in clusters for the company. The results also show that using the CC perturbation does not increase the time needed to find the incumbent (while it helps improving the solution quality, see Table 2), and in some cases it consistently reduces it (e.g., for Forli, Pescara, and Ravenna).

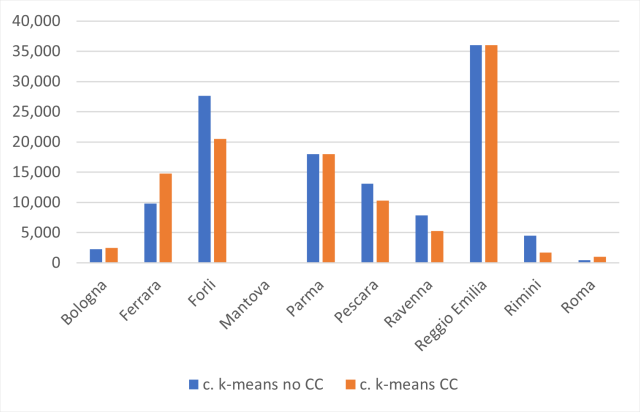

Figure 2: Computational times (in seconds) required to find the incumbent solutions by the ILS with the constrained k-means.

6 CONCLUSIONS AND FUTURE RESEARCH

This study has addressed a real-world car patrolling application that a large Italian service company faces. The problem consists of

Table 1: Solution values obtained by the constructive algorithm under different initial cluster configurations.

Instance	#	$\|C\|$	n	company	c. k-means	k-medoids	p-center	p-median
Bologna	8	239	2733	**40601.1**	21067.4	36549.2	32250.9	33616.1
Ferrara	7	236	1812	**44781.5**	42198.4	40363.8	40363.8	38935.6
Forli	8	407	2695	58562.2	**65736.4**	40050.6	37824.8	48803.9
Mantova	2	43	171	3423.1	3054.1	**3440.9**	**3440.9**	**3440.9**
Parma	5	289	2952	62982.7	**63996.1**	31502.7	39160.6	45928.9
Pescara	4	227	1370	32302.2	31306.8	**32390.0**	31726.1	32248.9
Ravenna	5	172	1382	28755.2	29340.6	29263.1	**29355.3**	29176.5
Reggio Emilia	10	679	7812	95254.6	**109834.9**	92291.8	93063.4	96162.4
Rimini	4	154	987	17472.1	17369.0	**17728.2**	17615.4	17167.4
Rome	8	118	1234	19164.7	19633.2	**20607.1**	19050.8	19586.4
AVG solution value				40329.9	**40353.7**	34418.7	34385.2	36506.7

Table 2: Solution values obtained by different ILS configurations.

				company		c. k-means		k-medoids		p-center		p-median	
Instance	#	$\|C\|$	n	no CC	CC	no CC	CC	no CC	CC	no CC	CC	no CC	CC
Bologna	8	239	2733	48929.9	**49750.9**	34311.8	34311.8	47429.5	48191.7	42175.5	39189.7	41613.4	42075.2
Ferrara	7	236	1812	50711.5	**50727.7**	48704.4	49143.5	44526.3	48958.0	44581.3	43696.9	43346.0	44544.6
Forli	8	407	2695	67369.3	67191.5	73298.8	**73345.5**	44890.7	45513.8	41436.6	50548.3	53654.9	48906.5
Mantova	2	43	171	4407.1	4407.1	4323.7	4323.7	**4408.8**	**4408.8**	**4408.8**	**4408.8**	**4408.8**	**4408.8**
Parma	5	289	2952	71269.6	69654.7	70517.7	**72145.3**	32143.4	58208.7	40977.9	39647.8	49228.5	37303.1
Pescara	4	227	1370	36956.3	36903.2	36384.9	36373.7	37392.4	37761.4	35693.8	37553.9	36734.7	**38354.3**
Ravenna	5	172	1382	**33318.0**	33199.7	33088.4	33180.1	32787.5	29410.6	33172.5	31989.0	32892.4	32404.3
Reggio Emilia	10	679	7812	110395.9	111988.7	124023.7	**127121.4**	118746.8	112177.4	92050.4	93541.9	103730.4	97987.5
Rimini	4	154	987	**20491.5**	20468.9	20309.4	20213.1	20254.1	20109.5	20161.3	20133.3	20246.1	19020.7
Rome	8	118	1234	25175.9	25162.9	24983.7	24988.9	25456.8	25418.8	23957.6	**25742.3**	25294.7	25665.2
AVG solution value				46902.5	46945.5	46994.7	**47514.7**	40803.6	43015.9	37861.6	38645.2	41115.0	39067.0
AVG impr. w.r.t. constructive				19%	19%	23%	23%	18%	26%	15%	17%	16%	12%

planning the routes that a fleet of patrols has to perform to deliver different security services to private customers. Each patrol takes care of a cluster of customers, serving them on a weekly basis. Not all services can be performed, but a minimum service level must be imposed to guarantee that a minimum percentage of services is satisfied.

The resulting optimization problem is a very challenging variant of the team orienteering problem that includes complicating additional constraints (especially the one on interdependent time windows). Due to its strong NP-hardness, we have chosen to solve it through a metaheuristic approach. In more detail, we have developed a two-step algorithm composed of clustering and routing subproblems. Several methods have been tested for the clustering subproblem, including some classical algorithms from the machine learning field and some MILP models. An ILS equipped with an inner VND was proposed for the routing component. An extra intra-cluster neighborhood structure was also included in the ILS to enable changes in the initial set of clusters, which is performed during the ILS iterations.

Our algorithm was tested on real-world instances that the company provided. The results showed an improvement for eight out of ten instances concerning a previous ILS that did not consider cluster modifications, showing that the proposed methodology can be very well suited to the context of the strategic company's operations. The only concern arising from our results is that such improvements are scattered across our several clustering algorithms. We have thus advised the company to use the algorithm in a multi-start way since the initial clustering is to be done only once. Then, as future research, we plan to produce a unique algorithm that gathers together the best aspects of the different components we tested.

For future studies, we also intend to extend this work to include dynamic stochastic features in our routing algorithm. This will allow it to consider the possible triggers of alarms that patrols must check upon demand. The resulting algorithm should run with low computing times so as to be used dynamically. Another topic for future studies is modeling the minimum time between two visits to the same customer for the same service as a soft constraint. This would be obtained by imposing an additional penalty in the objective function, and it could enable the solution algorithm to consider a wider set of solutions.

ACKNOWLEDGMENTS

This work was partially supported by: JSPS KAKENHI Grant Numbers JP20H02388, JP23K20268 and JP22H00513; NRRP, Mission 04 Component 2 Investment 1.5, Call 3277, Award 0001052; and Coopservice SCPA.

REFERENCES

[1] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. 2020. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics* 9, 8 (2020), 1295.

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

[3] Subhodip Biswas, Fanglan Chen, Andreea Sistrunk, Sathappan Muthiah, Zhiqian Chen, Nathan Self, Chang-Tien Lu, and Naren Ramakrishnan. 2020. Geospatial clustering for balanced and proximal schools. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 13358–13365.

[4] Paul S Bradley, Kristin P Bennett, and Ayhan Demiriz. 2000. Constrained k-means clustering. *Microsoft Research, Redmond* 20 (2000).

[5] John Gunnar Carlsson and Erick Delage. 2013. Robust partitioning for stochastic multivehicle routing. *Operations research* 61, 3 (2013), 727–744.

[6] Karl F Doerner, Manfred Gronalt, Richard F Hartl, Guenter Kiechle, and Marc Reimann. 2008. Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows. *Computers & Operations Research* 35, 9 (2008), 3034–3048.

[7] Forbes. 2017. Private Security Outnumbers The Police In Most Countries Worldwide [Infographic]. https://www.forbes.com/sites/niallmccarthy/2017/08/31/private-security-outnumbers-the-police-in-most-countries-worldwide-infographic/?sh=b2e7cd8210fb. [Online; Accessed 26 January 2024].

[8] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. 2014. A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics* 20, 3 (2014), 291–328.

[9] Bruce L Golden, Larry Levy, and Rakesh Vohra. 1987. The orienteering problem. *Naval Research Logistics (NRL)* 34, 3 (1987), 307–318.

[10] Aldy Gunawan, Hoong Chuin Lau, and Kun Lu. 2015. *Well-Tuned ILS for Extended Team Orienteering Problem with Time Windows*. Technical Report LARC-TR-01-15. Living Analytics Research Center, Available at http://research.larc.smu.edu.sg/larcweb/larc/publications/technicalreports/Well-Tuned-ILS-for-Extended-Team-Orienteering-Problem-with-Time-WindowsTR-01-15.pdf.

[11] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. 2016. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research* 255, 2 (2016), 315–332.

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

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

[14] M Khodadadian, A Divsalar, C Verbeeck, A Gunawan, and P Vansteenwegen. 2022. Time dependent orienteering problem with time windows and service time dependent profits. *Computers and Operations Research* 143, March (2022), 105794.

[15] Serhan Kotiloglu, Theodoros Lappas, Konstantinos Pelechrinis, and PP Repoussis. 2017. Personalized multi-period tour recommendations. *Tourism Management* 62 (2017), 76–88.

[16] IS Litvinchev, G Cedillo, and M Velarde. 2017. Integrating territory design and routing problems. *Journal of Computer and Systems Sciences International* 56 (2017), 969–974.

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

[18] Juan G Moya-García and M Angélica Salazar-Aguilar. 2020. Territory design for sales force sizing. *Optimal districting and territory design* (2020), 191–206.

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

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[21] LKPJ Rdusseeun and P Kaufman. 1987. Clustering by means of medoids. In *Proceedings of the statistical data analysis based on the L1 norm conference, neuchatel, switzerland*, Vol. 31.

[22] Roger Z Ríos-Mercado. 2020. *Optimal districting and territory design*. Vol. 284. Springer.

[23] Roger Z Ríos-Mercado and Juan C Salazar-Acosta. 2011. A GRASP with strategic oscillation for a commercial territory design problem with a routing budget constraint. In *Advances in Soft Computing: 10th Mexican International Conference on Artificial Intelligence, MICAI 2011, Puebla, Mexico, November 26-December 4, 2011, Proceedings, Part II 10*. Springer, 307–318.

[24] Ana M Rodrigues and J Soeiro Ferreira. 2015. Sectors and routes in solid waste collection. In *Operational Research: IO 2013-XVI Congress of APDIO, Bragança, Portugal, June 3-5, 2013*. Springer, 353–375.

[25] Sukanya Samanta, Goutam Sen, and Soumya Kanti Ghosh. 2021. A literature review on police patrolling problems. *Annals of Operations Research, 316* (2021), 1063–1106.

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

[27] Michael Schneider, Andreas Stenger, Fabian Schwahn, and Daniele Vigo. 2015. Territory-based vehicle routing in the presence of time-window constraints. *Transportation Science* 49, 4 (2015), 732–751.

[28] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)* 42, 3 (2017), 1–21.

[29] Theodore Tsiligirides. 1984. Heuristic methods applied to orienteering. *Journal of the Operational Research Society* 35, 9 (1984), 797–809.

[30] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. 2011. The orienteering problem: A survey. *European Journal of Operational Research* 209, 1 (2011), 1–10.

[31] Victor Hugo Vidigal Corrêa, Hang Dong, Manuel Iori, André Gustavo dos Santos, Mutsunori Yagiura, and Giorgio Zucchi. 2024. An iterated local search for a multi-period orienteering problem arising in a car patrolling application. *Networks* 83 (2024), 153–168.

[32] A Villalba and ECGL Rotta. 2022. Clustering and heuristics algorithm for the vehicle routing problem with time windows. *International Journal of Industrial Engineering Computations* 13, 2 (2022), 165–184.

[33] Richard T Wong. 2008. Vehicle routing for small package delivery and pickup services. In *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, 475–485.

[34] Wenzheng Xu, Weifa Liang, Zichuan Xu, Jian Peng, Dezhong Peng, Tang Liu, Xiaohua Jia, and Sajal K. Das. 2021. Approximation algorithms for the generalized team orienteering problem and its applications. *IEEE/ACM Transactions on Networking* 29, 1 (2021), 176–189.

[35] Shu Zhang, Jeffrey W Ohlmann, and Barrett W Thomas. 2020. Multi-period orienteering with uncertain adoption likelihood and waiting at customers. *European Journal of Operational Research* 282, 1 (2020), 288–303.

[36] Lin Zhou, Lu Zhen, Roberto Baldacci, Marco Boschetti, Ying Dai, and Andrew Lim. 2021. A Heuristic Algorithm for solving a large-scale real-world territory design problem. *Omega* 103 (2021), 102442.