UNIVERSITY OF MODENA AND REGGIO EMILIA
Department of Sciences and Methods for Engineering
Doctorate School in Industrial Innovation Engineering
XXXVii Cycle

Car patrolling problem: Algorithms for a real-world application

DOCTORAL THESIS

Author:
Victor Hugo Vidigal CORRÊA

Supervisor:
Prof. André Gustavo DOS SANTOS
Co-Supervisor:
Prof. Manuel IORI

Academic year 2023/2024

UNIVERSITY OF MODENA AND REGGIO EMILIA

Abstract

Doctorate School in Industrial Innovation Engineering
Department of Sciences and Methods for Engineering

Doctor of Philosophy

Car patrolling problem: Algorithms for a real-world application

by Victor Hugo Vidigal CORRÊA

This Ph.D. thesis addresses a Car Patrolling Problem derived from a real-world application faced by a large Italian service provider through the development of optimization algorithms.

In summary, the car patrolling problem that we face is a generalization of the well-known team orienteering problem. A set of customers requires a variety of security-related services that they want to be performed in their properties according to a weekly planning. On the basis of the customers' demands, the company deploys patrols that will perform the requested services. The goal of the research is to optimize and improve the service provision using algorithmic methods, possibly improving both customers satisfaction and reducing company costs. The research was conducted sequentially as requirements appeared during its development, and hence, the overall original problem was divided into three problems.

The first problem addresses patrol routing within their original territories, modeled as a deterministic variant of the team orienteering problem, where not all vertices need to be visited, and visiting them may yield profit. It was formalized using integer linear programming, and a metaheuristic algorithm based on Iterated Local Search (ILS) was developed to effectively solve the company's real-world instances.

Next, we addressed territory division, extending the previous problem by proposing new customer partitions to improve patrol routing. We explored methods based on clustering algorithms and Mixed Integer Linear Programming (MILP) models to generate alternative territory divisions. For each clustering solution, routing was performed using variants of the MILP model and clustering algorithms with the ILS from the first problem. Computational results showed that adjusting territory configurations can enhance customer satisfaction and reduce costs.

The third problem introduces stochastic elements in the form of alarm triggers within the patrolling operation. By employing a two-phase stochastic modeling approach, scenarios are generated based on an extensive historical alarm database to design resilient routes. The problem is addressed using both MILP and a heuristic method, delivering a comprehensive solution that enhances decision-making efficiency for managers.

Overall, this thesis has positively addressed a very general real-world problem with numerous practical challenges, providing efficient optimization algorithms and leading to improved problem solutions.

UNIVERSIDADE DE MODENA E REGGIO EMILIA

Resumo

Escola de Doutorado em Engenharia de Inovação Industrial
Departamento de Ciências e Métodos de Engenharia

Doutor em Filosofia

Car patrolling problem: Algorithms for a real-world application

por Victor Hugo Vidigal CORRÊA

Esta tese de doutorado aborda o problema de patrulhamento de carros derivado de uma aplicação real enfrentada por um grande provedor de serviços italiano, por meio do desenvolvimento de algoritmos de otimização.

Em resumo, o problema de patrulhamento de carros enfrentado é uma generalização do conhecido problema de orientação de equipe (team orienteering problem). Um conjunto de clientes requer uma variedade de serviços relacionados à segurança, que devem ser realizados em suas propriedades de acordo com um planejamento semanal. Com base nas demandas dos clientes, a empresa aloca patrulhas para realizar os serviços solicitados. O objetivo da pesquisa é otimizar e melhorar a prestação dos serviços utilizando métodos algorítmicos, buscando aumentar a satisfação dos clientes e reduzir os custos da empresa. A pesquisa foi conduzida de forma sequencial, conforme os requisitos surgiam durante o desenvolvimento, e, por isso, o problema original foi dividido em três subproblemas.

O primeiro subproblema aborda o roteamento das patrulhas dentro de seus territórios originais, modelado como uma variante determinística do problema de orientação de equipe, em que nem todos os vértices precisam ser visitados, e visitá-los pode gerar lucro. Esse subproblema foi formalizado utilizando programação linear inteira e, para resolvê-lo de forma eficiente em instâncias reais da empresa, foi desenvolvido um algoritmo metaheurístico baseado em Iterated Local Search (ILS).

Em seguida, abordamos a divisão de territórios, estendendo o problema anterior ao propor novas partições de clientes para melhorar o roteamento das patrulhas. Exploramos métodos baseados em algoritmos de agrupamento (clustering) e modelos de Programação Linear Inteira Mista (MILP) para gerar divisões alternativas de territórios. Para cada solução de agrupamento, o roteamento foi realizado utilizando variantes do modelo MILP e algoritmos de clustering com o ILS do primeiro subproblema. Os resultados computacionais demonstraram que ajustar as configurações dos territórios pode melhorar a satisfação dos clientes e reduzir os custos.

O terceiro subproblema introduz elementos estocásticos na forma de alarmes disparados durante as operações de patrulhamento. Utilizando uma abordagem de modelagem estocástica em duas fases, os cenários foram gerados com base em um extenso banco de dados histórico de alarmes para projetar rotas resilientes. O problema foi abordado utilizando tanto modelos MILP quanto um método heurístico, oferecendo uma solução abrangente que melhora a eficiência na tomada de decisões pelos gestores.

No geral, esta tese abordou de forma positiva um problema real muito amplo, com inúmeros desafios práticos, fornecendo algoritmos de otimização eficientes e levando a soluções aprimoradas para o problema.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Efficient security and asset protection are critical challenges faced by organizations and governments worldwide. In environments where mobile resources, such as patrol cars, are deployed to ensure safety, effective planning is crucial to optimize operations while minimizing costs.

Driven by the needs of a major Italian service provider, this thesis investigates a practical and impactful application of the CPP. The problem combines elements of vehicle routing, territorial division, and uncertainty, reflecting the complexities encountered in real-world operations. While existing solutions often address individual aspects of the problem, comprehensive approaches that integrate these dimensions remain limited.

Patrolling operations involve multiple interrelated challenges. First, companies must determine how to deploy patrol units efficiently, ensuring that all critical locations receive adequate coverage while minimizing operational costs. Second, the territory must be structured in a way that allows patrols to become familiar with their assigned areas, improving response times and service quality. Finally, real-world uncertainties—such as alarm-triggering events—introduce an additional layer of complexity, requiring solutions that can dynamically adapt to changing conditions.

To illustrate these challenges, Figure 1.1 presents a real-world instance from the Parma province. The nodes represent customer locations requiring periodic visits, while the color-coded clusters highlight territorial divisions for patrol assignments. The problem, therefore, is not only about routing but also about strategic territory design and handling unexpected events in an efficient manner.

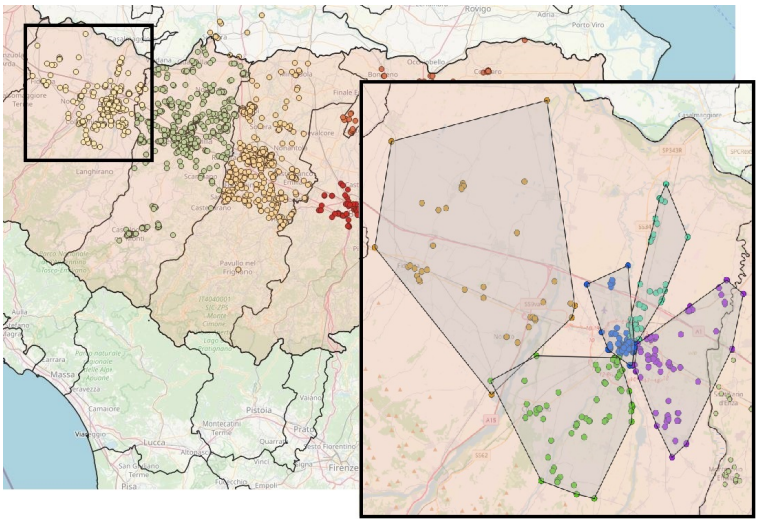


Figure 1.1: Customers in Emilia Romagna region with a more detailed look in Parma province, where customers are divided in clusters

The CPP emerges as a vital component of this broader domain, involving the strategic deployment of patrol units to maximize coverage and timely response to incidents. The core component of the CPP is the vehicle routing problem, discussed in detail in Chapter 2. However, the CPP contains unique features that distinguish it from standard formulations.

First, the CPP includes a set of optional services that, when performed, award a score. This aspect aligns it closely with the Orienteering Problem (OP) [64], a generalization of the Vehicle Routing Problem (VRP) [13]. However, unlike the OP, some services are mandatory, retaining key characteristics of the VRP. Furthermore, additional operational constraints add to the complexity of the problem. For instance, services can be performed multiple times within a single shift, but a minimum time interval must separate consecutive executions (i.e., we have interdependent time windows). Additionally, the company mandates that the overall quality of service meets a minimum threshold, adding another layer of difficulty to the optimization process.

In Chapter 2, we present a mathematical model to represent the CPP and an Iterated Local Search (ILS) algorithm [41] to solve it. Through extensive computational experiments, we demonstrate that the proposed algorithm significantly improves upon the company's previous solution methods, providing more effective and reliable outcomes for their operations.

In Chapter 3, we address another critical component of the CPP: territory division. The provinces where the company provides patrolling services are organized into distinct regions. This structure is essential, as patrols become familiar with their assigned territories over time, ultimately improving the quality of service. To enhance operational efficiency, we propose a combination of methods to redefine existing territories and generate new ones for potential areas where the company may expand its operations, making this a vital strategic element.

For this purpose, we develop a two-stage algorithm. In the first stage, customer clustering is performed using four different approaches: two classical clustering methods, namely k-Means [12] and k-Medoids [50], and two integer linear programming (ILP) models, namely p-Medians [51] and p-Centers [36]. In the second stage, we introduce a novel operator, building on our previous ILS framework, which allows customers to be reassigned to different patrols. This adjustment results in redesigned territories that enhance patrolling efficiency by creating more cohesive and manageable regions.

The algorithm is modular, allowing it to be tailored to specific managerial requirements. A series of experiments demonstrate that the proposed algorithm effectively achieves its objectives, offering significant improvements in both existing and newly defined territories.

The final component of the CPP involves addressing stochastic routing. During a work shift, the company must respond to alarm occurrences, which are unpredictable and only known at the moment they arise. As a result, deterministic optimization methods cannot adequately handle this feature. In Chapter 4, we introduce a two-stage mathematical model and a simheuristic to tackle this challenge.

The mathematical model provides a deterministic framework that represents the problem using a set of scenarios, each reflecting a possible realization of the stochastic variables. Meanwhile, the simheuristic employs a simulation-based approach, incorporating a recourse function to evaluate how a deterministic solution performs across stochastic scenarios [35]. This evaluation guides the heuristic, focusing on improving the solution's performance in stochastic realizations rather than optimizing the deterministic case alone.

Additionally, this chapter investigates the critical aspect of scenario generation. Scenarios are derived from alarm predictions provided by the company [1]. We propose a tree-like algorithm with adjustable parameters to create diverse scenario samples, enabling a flexible and robust representation of the uncertainty inherent in the problem.

In this thesis, we address a complex real-world problem by developing practical solution methods that can be reliably used in the daily operations of a patrolling company. The chapters of this thesis are organized as follows: Chapter 2 introduces the routing component of the problem, which serves as the foundation for the entire study. Chapter 3 adopts a more strategic perspective on the Car Patrolling Problem (CPP), aiming to improve routing operations not only through optimized routes but also by reorganizing patrolling territories. Finally, Chapter 4 investigates the stochastic nature of the CPP, incorporating alarms—unpredictable events that must be considered in the final routing plan. Together, these contributions not only provide a robust framework for patrolling operations but also offer insights applicable to any problem that can be modeled similarly.

Chapter 2

An Iterated Local Search for a Multi-period Orienteering Problem Arising in a Car Patrolling Application*

2.1 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 for 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/). With more than 25 000 employees, Coopservice offers various services, including logistics, transportation, cleaning, maintenance and security. The company also offers car patrolling services all over Italy for customers who booked their service. Figure 2.1 shows the current customers of the Emilia Romagna region, divided by province.

The customers are geographically dispersed in the area and are consequently divided into clusters. Each cluster is assigned to a patrol, which performs every working shift (e.g., eight hours) a route to visit customers and execute the required services. Figure 2.2 provides better details for the province of Reggio Emilia, showing the customers divided into patrolling clusters. The cluster configuration does not change from one 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 by 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

*The results of this chapter appears in: Vidigal Corrêa, V. H., Dong, H., Iori, M., dos Santos, A. G., Yagiura, M., & Zucchi, G. (2024). An iterated local search for a multi-period orienteering problem arising in a car patrolling application.". In: *Networks, 83(1), 153-168.*[17]

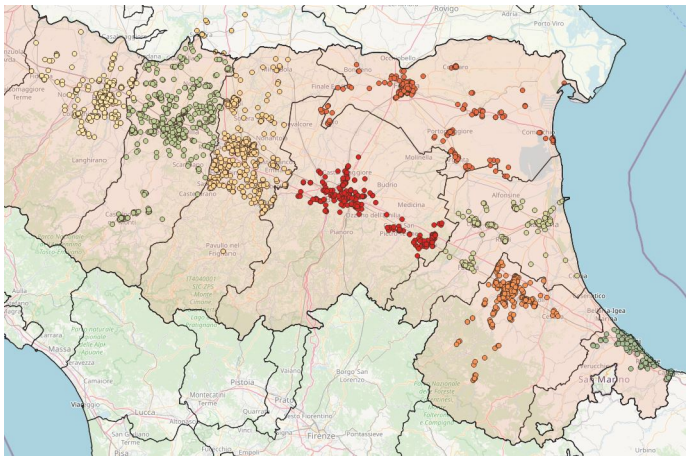


Figure 2.1: Customers in the Emilia Romagna region, divided by province

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.

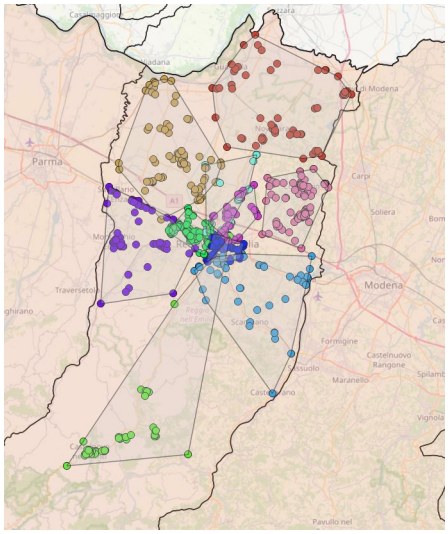


Figure 2.2: Customers in the Reggio Emilia province, divided by clusters

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

Our goal is to determine a set of routes, one per period, by optimizing an objective function that takes into account the total collected score and the total working time. The resulting problem is a multi-period orienteering problem, which is a generalization of the well-known orienteering problem (OP) [25]. 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 [42]. The ILS receives in input the set of customers and the set of services to be performed. It first builds an initial solution by means of a constructive heuristic. Then, as long as termination conditions are not met, it iteratively applies perturbations on the current incumbent solution and looks for improvements by means of a variable neighborhood descent (VND) procedure [31, 32] 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 2.2 contains a brief literature review of patrolling applications and OPs. Section 2.3 formally describes the problem. Section 2.4 presents the mathematical formulation. Section 2.5 gives the details of the ILS algorithm. Section 2.6 shows the computational results and, finally, Section 2.7 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 [73].

2.2 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 have been intensively studied in the literature, sometimes under different names, because they can model different applications. Very recently, [56, 57] 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 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 [64], 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, [24] 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 [29] and [65]. More in detail, [65] formally described the most relevant problem variants, and surveyed known exact and heuristics algorithms, whereas [29] computationally evaluated eight algorithms to solve the team orienteering problem with time windows, finding out that the ILS by [28] was the algorithm producing the best solutions on average. Surveys on related classes of problems were presented by [23], who focused on tourist trip design problems and by [6], 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 [47] 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 [3], who presented an integer linear stochastic model and solved it by branch-and-cut. Computational results were presented on instances containing up to 100 vertices.

A problem similar to ours, although with a different application, was studied by [39] under the name of personalized multi-period tour recommendation. The goal of the problem is to generate tours that include 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 [5]. In this problem, customers are grouped in clusters, and a profit is associated with each cluster. 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 to 1084 vertices.

In [30], 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 [71]. 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 [4], 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 [69], 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 [38]. 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.

2.3 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 $C_0 = \{0, 1, \ldots, n\}$, where 0 is the depot at which the single vehicle starts and ends each route, and $C = \{1, \ldots, n\}$ is the set of customers. The graph is complete and a traveling time γ'_{ij} is associated with each arc $(i, j) \in A_0$, 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 $\bar{n}_{cdt}$ be the number of services that have been actually performed in a solution. We define the quality of service (QoS) level as $Q = \sum_{c \in C, d \in D_c, t \in T_{cd}} \bar{n}_{cdt} / \sum_{c \in C, d \in D_c, t \in T_{cd}} n_{cdt}$. Index Q represents the ratio of services that have been performed in the entire set of periods and it should be greater than or equal to an input threshold value $Q_{\min}$.

Every service t required by a customer c in a period d is associated with a time window $[e_{cdt}, l_{cdt}]$. This defines the earliest and latest possible times to start the execution of each of the n_{cdt} services. The time window defines a hard constraint: late

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

For each period, a patrol starts its route at the depot, 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 $\tau = 1, \ldots, n_{cdt}$ be the index of the τth visit performed at customer c for service t in period d. Then, the score collected at visit τ is $w_{ct\tau}$ and is such that $w_{ct1} = w_{ct}$ and $w_{ct\tau} > w_{ct,\tau+1}$ for $\tau = 1, \ldots, n_{cdt} - 1$. In this way, the more visits for a given service are performed at a customer in a period, the more the score decreases and hence the first visits for other services and/or other customers become preferable to another visit for the same service at the same customer. This helps to achieve a balanced number of visits among customers and services.

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

2.4 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:

$$\max z = \quad \alpha \sum_{d\in D}\sum_{v\in V_d} w_{vd}y_{vd} - \beta \sum_{d\in D} s_{n+1,d} \tag{2.1}$$

$$\text{s. t.} \quad \sum_{j\in V_d^{n+1}} x_{0jd} = \sum_{i\in V_d^0} x_{i,n+1,d} = y_{0d} = y_{n+1,d} = 1 \quad d \in D \tag{2.2}$$

$$\sum_{i\in V_d^0} x_{ivd} = \sum_{j\in V_d^{n+1}} x_{vjd} = y_{vd} \quad v \in V_d, d \in D \tag{2.3}$$

$$\sum_{v\in V_{cdt}} y_{vd} = n_{cdt} \quad c \in C, d \in D_c, t \in M_{cd} \tag{2.4}$$

$$s_{id} + q_i + \gamma_{ij} - \mathcal{M}(1 - x_{ijd}) \le s_{jd} \quad i \in V_d^0, j \in V_d^{n+1}, d \in D \tag{2.5}$$

$$e_{id} - \mathcal{M}(1 - y_{id}) \le s_{id} \quad i \in V_d^0, d \in D \tag{2.6}$$

$$s_{jd} \le l_{jd} + \mathcal{M}(1 - y_{jd}) \quad j \in V_d^{n+1}, d \in D \tag{2.7}$$

$$s_{jd} - s_{id} \ge \delta_{\min} - \mathcal{M}(2 - y_{id} - y_{jd}) \quad i,j \in V_{cdt} : i < j, t \in T_{cd}, c \in C, d \in D \tag{2.8}$$

$$\frac{1}{n}\sum_{d\in D}\sum_{v\in V_d} y_{vd} \ge Q_{\min} \tag{2.9}$$

$$x_{ijd} \in \{0,1\} \quad i \in V_d^0, j \in V_d^{n+1}, d \in D \tag{2.10}$$

$$y_{vd} \in \{0,1\} \quad v \in V_d \cup \{0, n+1\}, d \in D \tag{2.11}$$

$$s_{id} \ge 0 \quad i \in V_d \cup \{0, n+1\}, d \in D. \tag{2.12}$$

The objective function (2.1) maximizes the weighted sum of total collected score minus total working time, multiplied by, respectively, α and β. Constraints (2.2) guarantee that each route starts and finishes at the depot. Constraints (2.3) guarantee route connectivity and also enforce the relation between variables x and y. Constraints (2.4) guarantee that all mandatory services are executed. Constraints (2.5), (2.6) and (2.7) 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 (2.8) guarantee that visits for the same service required by a customer are separated by at least $\delta_{\min}$ units of time. Constraint (2.9) imposes the minimum QoS. Constraints (2.10), (2.11) and (2.12) give the domain of the decision variables.

2.5 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 feasible solutions concerning all problem constraints, including the minimum QoS. The local search is performed by means of a VND, an algorithm that sequentially invokes local search procedures with a set of neighborhoods and finds a locally optimal solution for this set [32]. 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 below.

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
```

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

$$z(s) = \alpha \sum_{d \in D} \mathcal{S}(\sigma_d) - \beta \sum_{d \in D} \mathcal{T}(\sigma_d). \tag{2.13}$$

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 based on the start time of their time windows. A route is constructed for each period in two phases: first, the mandatory vertices are inserted sequentially, each at the end of the current route, in the order in which they were sorted provided that this preserves feasibility, that is, a vertex is appended only if the solution remains feasible; otherwise it is skipped (this is always feasible for the mandatory services in the instances provided by the company). Later, optional vertices are appended one by one in the solution in the sorted order, each at the end of the current route, whenever the resulting route is feasible. The two phases

invoke the procedure Append(σ_d, v) that first checks if inserting vertex v at the end of route σ_d is feasible, and then, if feasibility is confirmed, it returns the expanded route having v at its end; otherwise, it return the original route σ_d.

Algorithm 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
```

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.

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
```

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

- N_1 = Remove optional: Remove an optional service vertex from a route, thus trying to decrease the working time;

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

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

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

Perturbation procedure. The perturbation procedure is introduced to escape from the locally optimal solution obtained by the VND. Two inter-period neighborhoods are used to this aim. At each iteration, the perturbation procedure randomly selects two periods, d_1 and d_2, and then it invokes, alternatively, one of the two neighborhoods. The 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 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 2.3 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

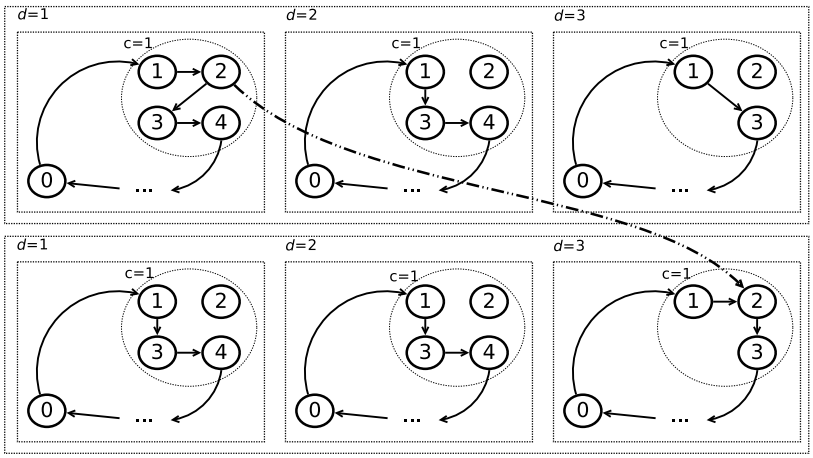


Figure 2.3: Depicting example for the Relocate inter-period perturbation movement

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.

2.6 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., [70]), 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 2.6.5 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.

2.6.1 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 2.1 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 2.1: 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

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

2.6.2 ILS results

Table 2.2 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 (2.1); 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 minutes. The average time was around 3 minutes or less for Mantova, Roma and Sassari, provinces requiring a small number of services, and larger than 50 minutes for Reggio Emilia, the province requiring the largest number of services. By comparing time_{inc} with time, we observe that the criteria adopted for the ILS termination are appropriate because the algorithm keeps running for some amount of time after the incumbent has been found, but this time is not excessive. 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 minutes, respectively. In the next sections, we obtain insights in the remaining KPIs by comparing them with the corresponding values obtained by the MILP model and by the company.

Table 2.2: 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

2.6.3 Comparison with the mathematical model

Table 2.3 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.

Table 2.3: 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

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

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

(c) Average quality of service (Q)

(d) Average traveled kilometers (km)

Figure 2.4: Four KPIs by Gurobi and ILS (on subset F of instances for which MILP found a feasible solution)

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 (2.9), which makes the model non-decomposable into single periods, and to constraints (2.5)–(2.8), 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 2.4, 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 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).

2.6.4 Comparison with the company solutions

In Table 2.4, 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.

Table 2.4: Comparison between company and ILS solutions on the full set of 79 instances

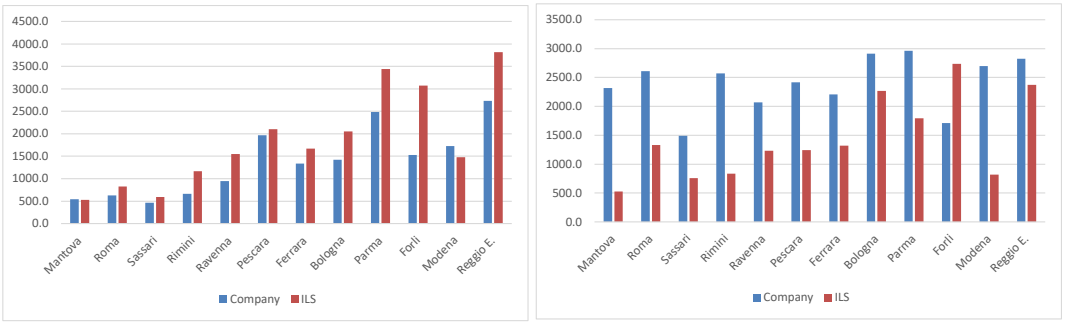
Instance		Company			ILS			
Province	#	z	Q	δ	z	Q	δ	impr.
Mantova	2	614.7	80.6	41.7	2164.7	82.6	92.3	252%
Roma	8	788.6	83.1	87.7	2945.6	82.2	98.9	274%
Sassari	7	1002.9	92.7	45.4	2941.7	89.1	94.4	193%
Rimini	4	1024.4	78.4	16.4	5093.6	92.2	91.5	397%
Ravenna	5	2884.7	90.8	51.9	6629.6	95.7	124.4	130%
Pescara	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%

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

We graphically compare in Figure 2.5 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.

2.6.5 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 2.5, which



(a) Average score ($\bar{s}$) **(b)** Average working time ($\overline{T}$)

Figure 2.5: Relevant KPIs for company and ILS (on the entire set of instances)

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 2.2.

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).

Table 2.5: 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%	-

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.

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 2.6. 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.

Table 2.6: 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

2.7 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.

Chapter 3

Optimizing a Car Patrolling Application by Iterated Local Search*

3.1 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 [22].

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 3.1 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

*The results of this chapter appears in: Corrêa, V. H. V., Alves de Queiroz, T., Iori, M., Dos Santos, A. G., Yagiura, M., & Zucchi, G. (2024, July). "Optimizing a Car Patrolling Application by Iterated Local Search". In: *In Proceedings of the Genetic and Evolutionary Computation Conference (pp. 1201-1209).*[17]

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 [66], 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.

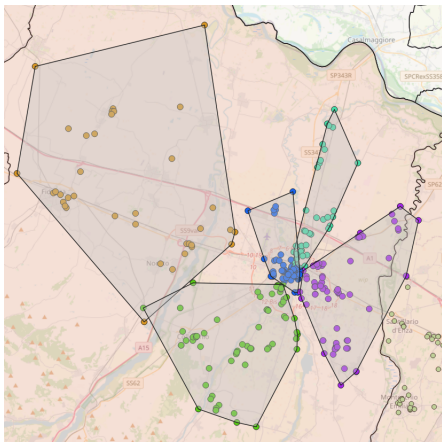


Figure 3.1: Customers in the Parma province, divided by clusters (solution adopted by the company).

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 [2], or based on density, like the Density-Based Spatial Clustering of Applications with Noise [59], 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 [25].

We solve the resulting problem by means of an ILS [42]. 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 [31, 32], 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 3.2, we delve into the literature on the problem and its two subproblems we deal with, namely clustering and routing. In Section 3.3, we formalize the overall problem. Then, in Section 3.4, we describe the solution methods we developed, and in Section 3.5, 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 3.6.

3.2 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, Sen, and Ghosh [56, 57], 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.

3.2.1 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 [64]. 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, Souffriau, and Oudheusden [65] and Gunawan, Lau, and Vansteenwegen [29]. In particular, Vansteenwegen, Souffriau, and Oudheusden [65] 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, Lau, and Vansteenwegen [29] 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, Lau, and Lu [28]. Other surveys concerning tourist trip design problems and vehicle routing problems were presented by, respectively, Gavalas et al. [23] and Archetti, Speranza, and Vigo [6].

Further literature contributions related to the problem we tackle in this work can be found in Palomo-Martínez et al. [47], Kotiloglu et al. [39], and Gündling and Witzel [30]. In Palomo-Martínez et al. [47], 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. [39], 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 [30]. 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, Ohlmann, and Thomas [71] 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. [69], 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. [38]. 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).

3.2.2 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 [68]. However, this approach may reduce route flexibility, mainly when integrating time-window constraints [58], so it must be tackled with care.

The literature on territory design (also referenced as districting/zone design, see, e.g., Ríos-Mercado [52]) is not limited to routing problems. Still, it includes, e.g., school redistricting [11] and sales force design [44]. Regarding routing problems, Ríos-Mercado and Salazar-Acosta [53] 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 [67] 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 [53] 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 [15] 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 [55], 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, Cedillo, and Velarde [40]. 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. [72] 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.

3.3 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. [66] 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 [19], 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

$$\max z = \alpha\mathcal{S} - \beta\mathcal{T}, \tag{3.1}$$

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

3.4 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 3.4.1) and then construct the routes for all clusters by an ILS (Section 3.4.2). While constructing the routes, we also attempt modifying the initial clusters to explore diverse solutions.

3.4.1 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 [12] and the k-medoids [50]. 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:

$$\min \quad \sum_{i \in C} \sum_{j \in C} \gamma_{ij} x_{ij} \tag{3.2}$$

$$\text{s. t.} \quad \sum_{i \in C} x_{ij} = 1 \quad \forall j \in C \tag{3.3}$$

$$\sum_{i \in C} y_i = p \tag{3.4}$$

$$\sum_{j \in C} x_{ij} \leq \Omega y_i \quad \forall i \in C \tag{3.5}$$

$$x_{ij} \in \{0,1\} \quad \forall i,j \in C \tag{3.6}$$

$$y_i \in \{0,1\} \quad \forall i \in C \tag{3.7}$$

The objective function (3.2) minimizes the total distance from each cluster center to each customer assigned to that cluster. Constraints (3.3) ensure that each customer is assigned to only one cluster. Constraints (3.4) ensure that exactly p clusters are opened. Constraints (3.5) 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 (3.6) and (3.7) 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:

$$\min \quad r \tag{3.8}$$

$$\text{s. t.} \quad r \geq \sum_{i \in C} \gamma_{ij} x_{ij} \quad \forall j \in C \tag{3.9}$$

$$\sum_{i \in C} x_{ij} = 1 \quad \forall j \in C \tag{3.10}$$

$$\sum_{i \in C} y_i = p \tag{3.11}$$

$$\sum_{j \in C} x_{ij} \leq \Omega y_i \quad \forall i \in C \tag{3.12}$$

$$x_{ij} \in \{0,1\} \quad \forall i,j \in C \tag{3.13}$$

$$y_i \in \{0,1\} \quad \forall i \in C \tag{3.14}$$

$$r \geq 0 \tag{3.15}$$

The objective function (3.8) minimizes the maximum distance. Constraints (3.10)–(3.14) are equivalent to constraints (3.3)–(3.7) above, respectively. Constraints (3.9)

and (3.15) state that r should be non-negative and not smaller than all selected distances.

3.4.2 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 [66], 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 4. 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 [32]. The ILS runs until a maximum run time or a maximum number of iterations without improvement is reached.

Algorithm 4 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
```

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 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 (3.1), 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 5.

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

Algorithm 5 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
```

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.

3.5 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 [49]. 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 3.1 the objective function values of the solutions obtained by the constructive heuristic (i.e., step 3 of Algorithm 4) when the different clustering methods of Section 3.4.1 are adopted. The solution values are computed using Equation (3.1). 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 3.1. 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 3.1 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 3.2 reports the objective function values of the solutions obtained with the ILS. We have again attempted the five territory-design configurations of Table 3.1, 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 $100(z_{ILS} - z_{constructive})/z_{constructive}$.

Table 3.1: 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

The results in Table 3.2 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

Table 3.2: 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%

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 3.2, 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 3.2), and in some cases it consistently reduces it (e.g., for Forli, Pescara, and Ravenna).

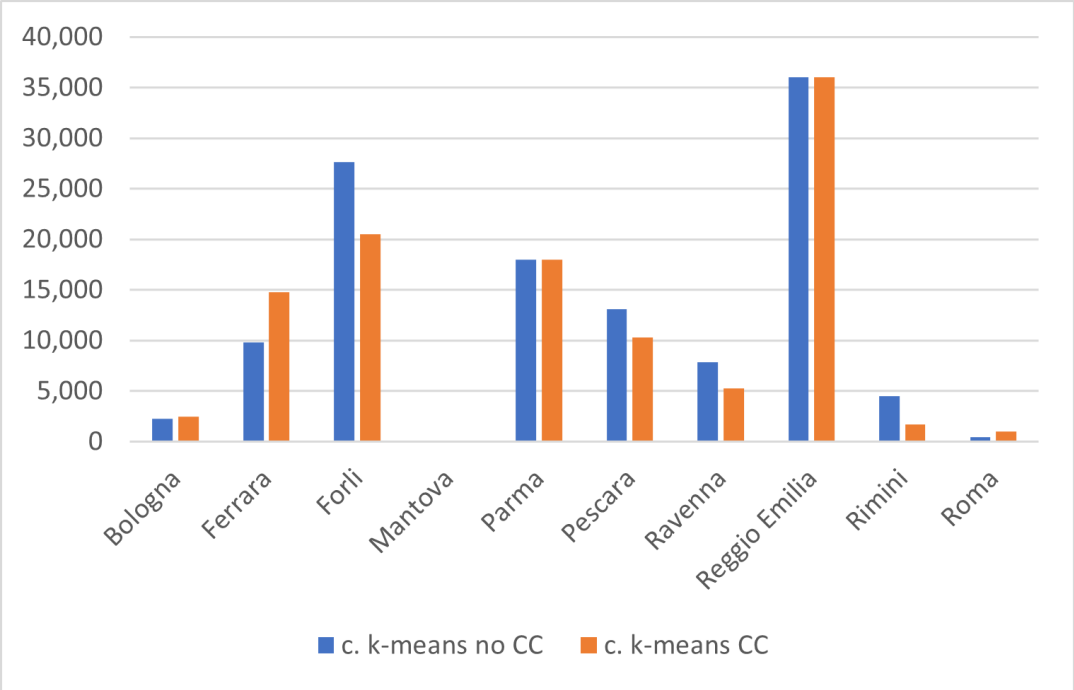


Figure 3.2: Computational times (in seconds) required to find the incumbent solutions by the ILS with the constrained k-means.

3.6 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 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.

Chapter 4

A scenario-based metaheuristic for the multi-period team orienteering problem with time windows and stochastic demands*

4.1 Introduction

Ensuring a secure environment is of great importance for business owners, as it is essential for protecting assets and maintaining uninterrupted operations. A fundamental component of security involves conducting regular patrols to monitor critical locations, deter potential threats, and facilitate rapid response to incidents. In many cases, the limited availability of public security measures require a significant role for the private security sector. This sector has experienced consistent growth, reaching a valuation of $180 billion in 2017, with projections indicating continued expansion in subsequent years. This trend is further underscored by data revealing that in half of the world's countries, the number of private security personnel exceeds that of public law enforcement officers [22].

From a managerial perspective, asset security comes in many forms, one of them being the patrolling. It consists of motorized or on-foot guards covering an area in which it is desired to reduce the chances of potential crimes that may disrupt business operations. Such type of service is provided by Coopservice, a major Italian service provider (https://www.coopservice.it/), that operates in the security market by providing patrolling services to customers across multiple Italian provinces. The business model of the company works as follows. Upon contract, interested business owners define a range of security related services that they want to be performed in their properties. With the demand for services provision set on a weekly basis, patrols will be designated to accomplish the demands in a defined territory throughout a working shift.

The problem, known as the Car Patrolling Problem (CPP), is structured as follows. At the provincial level, each province is divided into distinct patrolling territories, with a single vehicle assigned to each territory to address service demands. These demands encompass a variety of tasks, such as opening and closing gates, conducting internal patrols within buildings, and investigating potential criminal

*The results of this chapter appears in: Corrêa, V. H. V., Alves de Queiroz, T., Iori, M., Dos Santos, A. G., Yagiura, M., & Zucchi, G. (2024, December). "A scenario-based metaheuristic for the multi-period team orienteering problem with time windows and stochastic demands". Working paper, University of Modena and Reggio Emilia.[16]

activities. The services are classified as either mandatory or optional and must be performed within specified time windows.

Mandatory services are further classified based on their priority. Services with a defined priority index must adhere to a predetermined order of execution, while those without a priority index must still be completed but lack an explicit sequence. Mandatory services are divided into two types: (i) **ordinary mandatory services**, which resemble optional services but differ by being compulsory, and (ii) **uncertain alarm triggers**, which are not known in advance and occur probabilistically.

The customers who require the services are geographically dispersed in a large area and consequently divided into clusters. Each cluster is assigned a unique patrol, which performs one daily route to visit customers and execute the requested services. This is an important characteristic of how the company provides the patrolling service. Having patrols assigned consistently to the same cluster of customers allows them to perform the patrols more efficiently over time due to accumulated knowledge of the area.

Figure 4.1 illustrates the distribution of customers in the province of Parma, Italy, highlighting their division into clusters. While the cluster configuration remains fixed from day to day, the routes within each cluster may vary depending on the specific services required on a given day. These service requirements are determined by the contracts established with the company and can differ based on the day of the week. In previous research on this problem [66], the company's cluster configuration was treated as a fixed input, and the resulting routing problem was addressed independently within each cluster using an Iterated Local Search (ILS) approach. In subsequent research [16], we allowed for modifications to the cluster configurations, thereby integrating the multi-period routing component with the cluster design (also referred to as territory design), and exploring solutions over the entire set of clusters.

In this work, we extend the prior research by introducing a key advancement: accounting for uncertainties where some service requests, such as the verification of alarm triggers, may not be known in advance, leading to stochastic information. This extension aims to address the patrolling operation as a whole, ultimately improving the quality of service while addressing the increased complexity of the resulting stochastic problem.

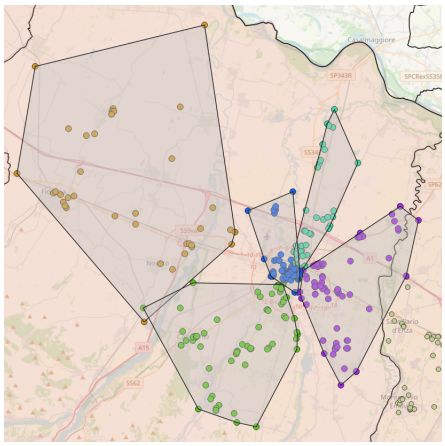


Figure 4.1: Customers in the Parma province, divided by clusters (solution adopted by the company).

The resulting optimization problem involves several operational constraints. Each customer may require multiple services and 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. If 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 routes, patrols may have to perform services that were not initially scheduled, like verifying the triggering of an alarm. When these happen, the patrol needs to verify them immediately, and upon completion, the patrol can continue with the other services.

This research aims to develop solutions methods for the problem englobing all the operational constraints presented. We tackle in two ways. First, a two-stage stochastic programming formulation of the problem is developed [7, 10]. The first stage comprises the routing of the deterministic part of the problem. The second stage composes the realization of the scenarios that are built over the stochastic data representing a probabilistic occurrence of alarms. Next, we present an algorithm based on the ILS metaheuristic. The algorithm counts with a recourse function to estimate the patrolling performance within scenarios. The estimated objective function value is then used to guide the algorithm towards our objective. This approach has been widely used in the literature to address complex stochastic problem where a deterministic solution has been previously developed [35].

The structure of this paper is as follows: Section 4.2 provides a review of the relevant literature on the problem. In Section 4.3, the overall problem is formally defined. Section 4.4 introduces the deterministic equivalent model derived from the two-stage stochastic programming formulation, offering a mathematical formalization of the problem. In Section 4.5, a scenario-based metaheuristic is proposed to address real-world instances. Section 4.6 presents the computational results obtained using the proposed model and metaheuristic. Finally, Section 4.7 offers concluding remarks and outlines directions for future research.

4.2 Literature review

The form of the car patrolling problem that we present is unique, and no other study has addressed it in this exact way. However, several studies in the literature relate to our problem, as they address its components either directly or indirectly. The immediate resemblance is with the vehicle routing problem (VRP) [18], where we must route a vehicle through a set of points. However, due to additional constraints, not all points require visits, and those that are visited award a score. This makes our problem similar to a VRP generalization known as the orienteering problem (OP) [25] while retaining core VRP characteristics. Additionally, certain demands in our problem are stochastic, with some points requiring service based on probabilistic events, further distinguishing it from classical VRP and OP formulations. Together, these characteristics place our problem within the class of NP-hard problems, making it challenging to address, especially for real-world instances.

The OP serves as a primary reference for our car patrolling problem due to its emphasis on selective visitation, where each chosen point offers a reward. In the OP, the objective is to maximize the accumulated score within constraints such as time or distance, aligning well with patrolling requirements where certain locations are optional but yield a benefit upon visitation. Several comprehensive reviews have been published on the OP [29, 65]. Specifically, [65] provides a formal description of the most significant problem variants and an overview of the existing exact and heuristic algorithms. Meanwhile, [29] presents a computational comparison of eight algorithms for the team orienteering problem with time windows, concluding that

the ILS proposed by [28] consistently produced the best average solutions. Additionally, surveys on related problem classes have been conducted by [23], focusing on tourist trip design problems, and by [6], discussing vehicle routing problems with profits.

Despite normally not requiring mandatory visits, few studies have incorporated this characteristic in the OP. A hybrid heuristic composed of a greedy randomized adaptive search procedure GRASPs and a variable neighborhood search (VNS) was proposed by [47]. 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, requiring, on average, only 0.8% of the time required by an mixed integer linear programming MILP model solved with a commercial solver. Continuing on this trend, a memetic algorithm was proposed by [43]. Utilizing a tabu search procedure aggregated in a genetic algorithm framework, the authors report improving 104 best known solutions for the set of 340 instances they use, and also report to have found 22 proven optimal solutions.

A novel variant of the Orienteering Problem (OP), incorporating service time-dependent profits and time-dependent travel times, was examined by [38]. The authors developed a Mixed-Integer Linear Programming (MILP) model and a Variable Neighborhood Search (VNS) metaheuristic, employing three specialized neighborhood structures: insertion, replacement, and swapping. The VNS was validated using a set of benchmark instances with known optimal solutions, achieving improvements or matches in 29 out of 36 cases. Subsequently, the approach was applied to a case study based on the city of Shiraz, Iran, where the VNS demonstrated the ability to produce high-quality solutions for real-world instances.

Tourism tour recommendation represents a recurrent application of the OP and shares several similarities with the CPP. In most instances of this type, visiting all points is not mandatory, and it is common to assign scores to visits to reflect user preferences. Motivated by this application, [28] proposed an Iterated Local Search (ILS) algorithm to address the problem. The algorithm begins with a greedy constructive heuristic that incrementally builds a solution by inserting visits into the route until a feasible solution is achieved. Perturbation is introduced by randomly removing visits from the solution, while local search is performed using swap and 2-opt operators. The algorithm was later extended in two subsequent studies [26, 27] with the incorporation of additional components, including new local search operators (swap1, swap2, 2-opt, move, insert, and replace) and hybridization with simulated annealing. Across a set of benchmark instances, the proposed method improved 31 best-known solutions and demonstrated suitability for real-world instances, producing high-quality solutions within reasonable computational times.

In the study presented by [39], the problem named personalized multi-period tour recommendation problem is addressed. The goal is to generate tours, including mandatory and optional visits, while maximizing the total collected score for the optional ones. Similar to the CPP, features like multiple periods of visits and time windows are considered. The problem is solved by means of iterated table search and is computationally evaluated by using two data sets, one from the literature and the other generated with real-world data. The algorithm uses several local search movements based on edge-exchange, namely intra-route and inter-route 2-opt, 1-0 relocate and 1-1 exchange. In the literature instance set, the algorithm produced several new best known solutions; while in the real-world instances, the algorithm was capable to generate tours respecting all the constraints of the problem.

In another study by [30], another tourism route recommendation application is addressed by means of MILP model and ILS metaheuristic. The goal is to optimize touristic routes, considering constraints such as visit redundancy avoidance and time windows. 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. This is a recurrent observation, when it comes down to OP applications. Model are used to address smaller instances and to access heuristic's reliability.

Two-stage heuristics have been effectively applied by leveraging problem decomposition and sequential resolution of interdependent components. For instance, [71] addresses a multi-period orienteering problem where a salesperson must visit a subset of customers within their time windows to maximize expected sales from product adoption. In their proposed two-stage heuristic, the first stage selects the customers to be visited, while the second stage solves a vehicle routing problem for the chosen subset. This approach aligns with the decision-making process of the salesperson upon arriving at customer sites and is validated using an adapted dataset from the vehicle routing problem with time windows. Similar applications of two-stage heuristics can be found in [63] and [34]. [63] employs a two-stage approach to solve the VRP logistics distribution. The problem is divided in two stages where the first consists in the distribution region partitioning and the second the routing phase. It is solved by hybrid genetic beam search algorithm. Likewise, [34] addresses a service scheduling problem, where the first stage prioritizes tasks, and the second stage allocates resources to minimize operational costs. Both studies demonstrate the flexibility and effectiveness of two-stage heuristics in decomposing complex combinatorial problems.

Regarding the stochastic aspects of our problem, several comprehensive studies have reviewed stochastic features in routing problems [8, 45, 46, 54]. Stochasticity in these problems can manifest in various forms. A consensus established by these surveys is that variants with stochastic demands are the most prevalent in the literature. This aligns with our problem, where certain demands are characterized probabilistically. Notably, [46] and [45] form a two-part survey that provides an in-depth examination of models and solution methods. These studies emphasize that both exact and heuristic methods are employed to address these problems, though heuristic methods are more frequently utilized, with local search-based approaches being particularly prominent. It is evident that these problems become increasingly challenging as additional constraints are introduced. Consequently, simpler solution methods have often been employed successfully, given their typically robust performance.

In the domain of stochastic orienteering problems, one can observe a concentration of studies focusing on stochasticity manifesting in collected profit (score) [33] and travel times [14, 21]. However, some studies relate to our by the name of probabilistic OP [3]. In this OP variant, a set of customers availability is given by probability. The goal is to maximize the difference between the total collected score and the total travelling time. The problem is dealt with by means of exact and heuristic algorithms, and the solution quality is evaluated by comparing how a deterministically obtained solution would behave in a stochastic scenario.

A systematical review has addressed a general framework to extend heuristics to deal with stochastic optimization problems [35]. The goal was to demonstrate that already existing algorithms for deterministic problems can be extended with minor additions to address their stochastic counterpart. The proposed framework is composed by a traditional heuristic that is incremented with a simulation method that is

capable to evaluate a deterministically generated solution in a set of scenarios. This is then used to guide the algorithm towards solutions that better suit the stochastic realizations. This approach has been successfully used in several applications [9, 37, 48].

4.3 Problem description

In this section, we formally describe the stochastic optimization problem we face, which we call *the multi-period team orienteering problem with time windows and stochastic demands* (MPTOPTWSD). We adopt the notation in [66] but extend it to the stochastic demand aspects. We are provided with a directed graph $G_0 = (C_0, A_0)$, where the set of nodes is $C_0 = \{0\} \cup C$. Here, 0 represents the depot from which all vehicles depart and to which they return, while $C = \{1, \ldots, |C|\}$ represents the set of customers. The graph is complete and asymmetrical, and each arc $(i, j) \in A_0$ is associated with a travel time γ_{ij}, where we impose $\gamma_{ii} = 0$ for all i. Also, every customer $c \in C$ is associated to a cluster $k \in K$ that represents a patrolling territory given as an input.

Let T denote the complete set of services provided by the company, which includes tasks such as closing or opening a commercial activity, inspecting a private residence, or responding to an alarm trigger. Each service $t \in T$ is associated with a standard service time q_t, representing the time required for a patrol to execute the service. The set of services is categorized into two primary types: mandatory services, denoted by M, and optional services, denoted by U, such that $T = M \cup U$. Additionally, a third category of services, represented by H, comprises alarm-triggering events. These services are also mandatory but are characterized by an associated random probability.

The activities are performed over a time span of D periods, where each period $d \in D$ corresponds to the working shift of a patrol. Each customer $c \in C$ requires services only during a subset $D_c \subseteq D$ of these periods and those should be provisioned by the same patrol, i.e., they should be part of the same cluster $k \in K$. Formally, we define $T_{cd} \subseteq T$ as the subset of services to be performed for customer c during period d, and we partition it as $T_{cd} = M_{cd} \cup U_{cd}$, where $M_{cd} \subseteq M$ represents the mandatory services and $U_{cd} \subseteq U$ denotes the optional services. Consequently, the number of visits required per period is not fixed and may vary from one period to another for each customer, so we define C_d as the subset of customers who have requested at least one service during period d. Furthermore, we define n_{cdt} as the number of times service t is required by customer c during period d, and $\bar{n}_{cdt}$ as the number of services actually performed in a given solution, with $0 \leq \bar{n}_{cdt} \leq n_{cdt}$. It is essential for the company to ensure that the quality of service meets a minimum standard. The quality of service (QoS) is defined as the ratio of services completed across all periods and clusters, given by

$$Q = \frac{\sum_{c \in C, d \in D_c, t \in T_{cd}} \bar{n}_{cdt}}{\sum_{c \in C, d \in D_c, t \in T_{cd}} n_{cdt}}.$$

To guarantee that the desired level of service is achieved, Q must be greater than or equal to a predefined threshold value $Q_{\min}$.

Additionally, set H contains the alarm events that may occur within a whole week period with a given probability and require attention. We formalize this by defining H_{cd} as the set of expected alarm occurrences for customer c during period d. We define the set Ω as the power set $2^H = \{S \mid S \subseteq H\}$ representing all possible scenario realizations. We consider that alarms triggered are independent; thus, the

probability of each scenario is set by the product of each alarm's probability in it, i.e., $P(\omega) = \prod_{h \in \omega} p(h)$. The number of scenarios is exponential; hence, we rely on a scenario reduction method that is described later.

Upon arrival at a customer, the patrol shall perform the requested services following some constraints. First, a service t required by customer c in period d is associated with a time window $[e_{cdt}, l_{cdt}]$, which represents the earliest and latest possible times to start the provision of the service. Arriving after l_{cdt} is not allowed, but if the patrol arrives before e_{cdt}, is allowed to wait until the time window opens to start the service provision. Alarms are not subjected to a formal time window, but rather they count with a triggering time. Also, a maximum response time is set and together with the triggering time compose a time window for the alarm $[e_{cdt}, l_{cdt}]$, where e_{cdt} is the triggering and l_{cdt} is e_{cdt} plus the desired response time.

Customers may require multiple visits 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 [19], and the threshold value serves to obtain a balanced patrol so that the visits are not too close.

The depot is also subjected to a time window represented by $[e_0, l_0]$. It has the purpose of imposing the maximum working time of each period and limit the maximum route duration to the shift duration. For 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).

Each service is assigned a priority b_{ct}, associated with a customer $c \in C$ and a service $t \in T$. Based on this priority, a weighted scoring function is employed to determine the score collected during each visit. The score is defined as $w_{ct\tau} = b_{ct} e^{1-\tau}$, where c and t represent the customer and the service, respectively, and τ denotes the number of the visit being performed. Specifically, the first visit collects the full score since $\tau = 1$ yields $w_{ct\tau} = b_{ct}$. For subsequent visits, as τ increases, the score collected decreases. This approach ensures that over time, high-priority services accrue a diminishing score, thereby increasing the likelihood of lower-priority services being performed. The objective is to achieve a more balanced and equitable allocation of services.

In summary, the MPTOPTWSD aims to construct a set of routes for each cluster and period, ensuring that all mandatory services are completed and that some or all optional services are performed within their respective, interdependent time windows. The routes must also account for the potential demand for alarm triggers, which are determined by a probability of occurrence. The objective is to maximize a weighted function that evaluates the expected realization of scenarios, weighted by their respective probabilities $P(\omega)$. This function incorporates the score $\mathcal{S}^{\omega}$, representing the services completed across all clusters and time periods, and subtracts the total working time $\mathcal{T}^{\omega}$ for the entire set of routes. Both components are scaled by their respective balancing factors, α and β. Formally, the objective function is defined as:

$$\max \bar{z} = \sum_{\omega \in \Omega} P(\omega)(\alpha \mathcal{S}^{\rightarrow} - \beta \mathcal{T}^{\rightarrow}) \tag{4.1}$$

where Ω denotes the set of scenarios.

4.4 Stochastic programming formulation

For the MPTOPTWSD, we developed a two-stage integer linear programming model [60]. The first stage is responsible for determining the routing configuration, which the second stage will use to address scenario realizations. We define an extended graph $G = (V_{dk}, A_{dk})$, in which each vertex represents a visit to a customer to perform a service. Specifically, let V_{dk} be the set of vertices representing all possible visits associated with the services requested by all customers in period $d \in D$ and cluster $k \in K$. Let $n = |\cup_{d \in D} V_{dk}|$ be the total number of vertices, noting that by construction, $n = \sum_{c \in C, d \in D, t \in T} n_{cdt}$. Additionally, let $V_{dk}^0 = V_{dk} \cup \{0\}$ and $V_{dk}^{n+1} = V_{dk} \cup \{n+1\}$, where 0 and $n+1$ are copies of the depot representing, respectively, the start and end of the route for each period $d \in D$. Hence, $V_{dk}^{0(n+1)} = V_{dk} \cup \{0\} \cup \{n+1\}$. The overall set of vertices in G is then defined as $V = \{0\} \cup \{\cup_{d \in D, k \in K} V_{dk}\} \cup \{n+1\}$. The graph G is further extended to accommodate the alarms in the scenarios Ω. For each $\omega \in \Omega$, we derive the set $V_{dk}^{\omega} = V_{dk} \cup \omega$. Figure 4.2 depicts the set V_{dk} and V_{dk}^{ω} for a single route, graph V_{00}, and three scenarios: V_{00}^0, V_{00}^1, and V_{00}^2. The graph V_{00} is the equivalent of V_{dk} for $d = 0$ and $k = 0$, while the following graphs represent V_{dk}^{ω} for three scenarios. The vertex d represents the depot, enumerated vertices represent the mandatory and optional services, and vertices labeled a represent alarms.

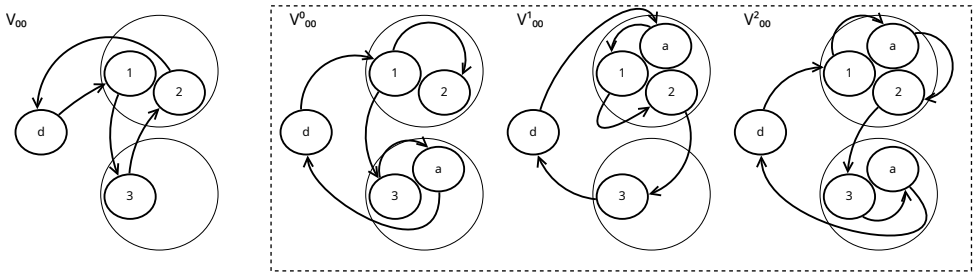


Figure 4.2: Model representation and an example solution.

We define V_{cdt} as the subset of V_{dk} representing the n_{cdt} visits for service $t \in T$ requested by customer $c \in C$ that belongs to cluster $k \in K$ 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 requested by customer c for this service. The graph is complete, asymmetrical, 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 $c = \hat{c}$).

As we develop a two-stage stochastic programming model, the first stage of the problem is related to decisions here-and-now involving the definition of the routing schedule disregarding the occurrences of alarms. In the second stage, wait-and-see decisions are made considering provision of alarms verifications. The objective maximizes the expected total collected score while minimizing the expected total working time.

For what concerns the scores, we use w_{vdk} to denote the score associated vertex $v \in V_{dk}$ in period $d \in D$ and cluster $k \in K$. The value of w_{vdk} 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 $c \in C_d$, $t \in T$, and $d \in D$. We also define ϵ as a punishment factor to be added in the objective function for mandatory services that are cancelled during the scenarios realizations.

The deterministic equivalent formulation of the two-stage stochastic problem has the following decision variables:

- x_{ijdk} takes value 1 if the patrol moves from vertex $i \in V_{dk}^{0}$ to vertex $j \in V_{dk}^{n+1}$ in period $d \in D$, cluster $k \in K$, 0 otherwise;
- x_{ijdk}^{ω} takes value 1 if the patrol moves from vertex $i \in V_{dk}^{\omega 0}$ to vertex $j \in V_{dk}^{\omega(n+1)}$ in period $d \in D$, cluster $k \in K$, scenario $\omega \in \Omega$, 0 otherwise;
- y_{vdk} takes value 1 if vertex $v \in V_{dk}^{0(n+1)}$ is visited in period $d \in D$, cluster $k \in K$, 0 otherwise;
- y_{vdk}^{ω} takes value 1 if vertex $v \in V_{dk}^{\omega(n+1)}$ is visited in period $d \in D$, cluster $k \in K$, scenario $\omega \in \Omega$, 0 otherwise;
- s_{vdk} gives the time at which the patrol arrives at vertex $v \in V_{dk}^{0(n+1)}$ in period $d \in D$, cluster $k \in K$;
- s_{vdk}^{ω} gives the time at which the patrol arrives at vertex $v \in V_{dk}^{\omega(n+1)}$ in period $d \in D$, cluster $k \in K$, scenario $\omega \in \Omega$;

The two-stage stochastic programming is defined with the objective function (4.2) and constraints (4.3)-(4.24):

$$\max \bar{z} = \sum_{\omega \in \Omega} P(\omega)(\alpha(\sum_{d \in D}\sum_{k \in K}\sum_{v \in V_{dk}^{\omega}} w_{vdk} y_{vdk}^{\omega} - \epsilon \sum_{d \in D}\sum_{k \in K}\sum_{v \in V_{dk}^{obr.}} w_{vdk}(y_{vdk} - y_{vdk}^{\omega})) - \beta \sum_{d \in D}\sum_{k \in K} s_{adk}^{\omega}) \tag{4.2}$$

$$\text{s. t.} \quad \sum_{j \in V_{dk}^{n+1}} x_{0jdk} = \sum_{i \in V_{dk}^{0}} x_{i(n+1)dk} = y_{0dk} = y_{(n+1)dk} = 1, \quad d \in D, k \in K \tag{4.3}$$

$$\sum_{i \in V_{kd}^{0}} x_{ivdk} = \sum_{j \in V_{dk}^{n+1}} x_{vjdk} = y_{vdk}, \quad v \in V_{kd}, d \in D, k \in K \tag{4.4}$$

$$\sum_{k \in K}\sum_{v \in V_{cdt}} y_{vdk} = n_{cdt}, \quad c \in C, d \in D_c, t \in M_{cd} \tag{4.5}$$

$$s_{idk} + q_i + \gamma_{ij} - \mathcal{M}(1 - x_{ijdk}) \leq s_{jdk}, \quad i \in V_{dk}^{0}, j \in V_{dk}^{n+1}, d \in D, k \in K \tag{4.6}$$

$$s_{idk} \geq e_{idk} - \mathcal{M}(1 - y_{idk}), \quad i \in V_{dk}^{0}, d \in D, k \in K \tag{4.7}$$

$$s_{jdk} \leq l_{jdk} + \mathcal{M}(1 - y_{jdk}), \quad j \in V_{dk}^{n+1}, d \in D, k \in K \tag{4.8}$$

$$s_{jdk} - s_{idk} \geq \delta_{\min} - \mathcal{M}(2 - y_{idk} - y_{jdk}), \quad i, j \in V_{cdt} : i < j, t \in T_{cdk}, \; c \in C, d \in D, k \in K \tag{4.9}$$

$$\frac{1}{n}\sum_{k \in K}\sum_{d \in D}\sum_{v \in V_{dk}} y_{vdk} \geq Q_{\min}, \tag{4.10}$$

$$\sum_{j \in V_{dk}^{\omega(n+1)}} x_{0jdk}^{\omega} = \sum_{i \in V_{dk}^{\omega 0}} x_{i(n+1)dk}^{\omega} = y_{0dk}^{\omega} = y_{(n+1)dk}^{\omega} = 1, \quad d \in D, k \in K, \omega \in \Omega \tag{4.11}$$

$$\sum_{i \in V_{kd}^{\omega 0}} x_{ivdk} = \sum_{j \in V_{dk}^{\omega(n+1)}} x_{vjdk} = y_{vdk}, \quad v \in V_{kd}^{\omega}, d \in D, k \in K, \omega \in \Omega \tag{4.12}$$

$$s_{idk}^{\omega} + q_i + \gamma_{ij} - \mathcal{M}(1 - x_{ijdk}^{\omega}) \leq s_{jdk}^{\omega}, \quad i \in V_{dk}^{\omega 0}, j \in V_{dk}^{\omega(n+1)}, d \in D, \; k \in K, \omega \in \Omega \tag{4.13}$$

$$s_{idk}^{\omega} \geq e_{idk} - \mathcal{M}(1 - y_{idk}^{\omega}), \quad i \in V_{dk}^{\omega 0}, d \in D, k \in K, \omega \in \Omega \tag{4.14}$$

$$s^{\omega}_{jdk} \leq l_{jdk} + \mathcal{M}(1 - y^{\omega}_{jdk}), \quad j \in V^{\omega(n+1)}_{dk}, d \in D, k \in K, \omega \in \Omega \quad (4.15)$$

$$s^{\omega}_{jdk} - s^{\omega}_{idk} \geq \delta_{\min} - \mathcal{M}(2 - y^{\omega}_{idk} - y^{\omega}_{jdk}), \quad i, j \in V_{cdt} : i < j, t \in T_{cdk}, \; c \in C, d \in D, k \in K, \omega \in \Omega \quad (4.16)$$

$$y^{\omega}_{idk} = 1, \quad i \in V^{\omega}_{kd} \cap H, d \in D, k \in K, \omega \in \Omega \quad (4.17)$$

$$y^{\omega}_{jdk} \leq y_{idk}, \quad i = j, i \in V_{kd}, j \in V^{\omega}_{kd}, d \in D, \; k \in K, \omega \in \Omega \quad (4.18)$$

$$x_{ijd} \in \{0,1\}, \quad i \in V^{0}_{d}, j \in V^{n+1}_{d}, d \in D \quad (4.19)$$

$$x^{\omega}_{ijd} \in \{0,1\}, \quad i \in V^{0}_{d}, j \in V^{n+1}_{d}, d \in D, \omega \in \Omega \quad (4.20)$$

$$y_{vd} \in \{0,1\}, \quad v \in V_d \cup \{0, n+1\}, d \in D \quad (4.21)$$

$$y^{\omega}_{vd} \in \{0,1\}, \quad v \in V_d \cup \{0, n+1\}, d \in D, \omega \in \Omega \quad (4.22)$$

$$s_{id} \geq 0, \quad i \in V_d \cup \{0, n+1\}, d \in D \quad (4.23)$$

$$s^{\omega}_{id} \geq 0, \quad i \in V_d \cup \{0, n+1\}, d \in D, \omega \in \Omega. \quad (4.24)$$

The objective function (4.2) is divided in two components. The first is the value associated with the average weighted sum of the total collected score. This is done by summing up the total collected score and later decrementing the score of mandatory services cancelled during the realization of scenarios. The second component is the total working time. They are both multiplied by, respectively, α and β. This is the second phase objective, where with the previous phase realized, we aim to optimize the routes according to the expectation criteria established by the company. The function is weighted by the probability of each scenario. Constraints (4.3) to (4.10) concerns the first stage. Constraints (4.3) and (4.4) imposes the route connectivity. While the former is responsible for depot nodes, the latter is responsible for services node. Constraints (4.5) impose that all mandatory services should be performed. Constraints (4.6) enforce the arrival time in each vertex while Constraints (4.7) and (4.8) impose time windows to be respected. Constraints (4.9) guarantee that visits for the same service required by a customer are separated by at least $\delta_{\min}$ units of time. Constraints (4.10) imposes the minimum QoS. Constraints (4.11) to (4.16) refer to the second stage and follow the same pattern from Constraints (4.3) to (4.9), with the difference that they are valid for each scenario composed by V^{ω}. Constraints (4.17) impose that all alarms should be checked. Constraints (4.18) connect the first stage with the second, stating that only services initially scheduled in the first stage can be performed in the second stage. Constraints (4.19) to (4.24) give the domain of the decision variables.

4.5 Scenario-based iterated local search

The Multi-Period Team Orienteering Problem with Time Windows and Stochastic Demands (MPTOPTWSD) is a generalization of the Orienteering Problem (OP), which is classified as $\mathcal{NP}$-hard. This classification indicates that our problem also belongs to this challenging complexity class. Additionally, the company's instances are quite large. For example, in the province of Reggio Emilia, where the company's headquarters are located, there are 679 customers and 7,812 services. Consequently, employing a metaheuristic algorithm is highly suitable for our needs.

Building on the simheuristic approach proposed in [35], we have extended our previous algorithm [17] to incorporate the operation of alarms as part of the optimization process. This method is based on a recourse algorithm capable of evaluating a deterministically generated solution against a selected set of scenarios. The deterministic algorithm then uses the expected objective function value to guide the

search accordingly. Figure 4.3 illustrates the scheme that enables metaheuristics to adapt to stochastic problems. By utilizing a scenario-based assessment before the acceptance criteria, it is possible to evaluate the solution's performance according to the realization of random variables, thereby guiding the algorithm towards improvement.

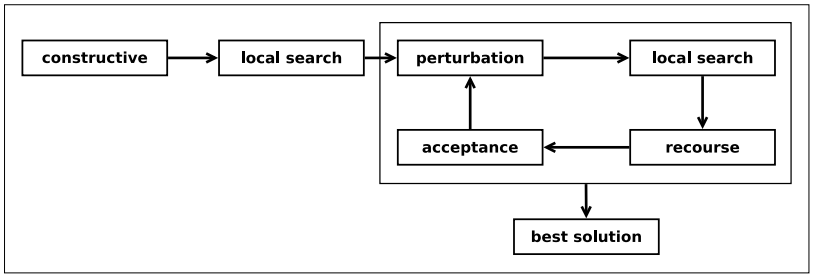


Figure 4.3: Depiction of the scenario based iterated local search

The proposed heuristic is derived from the ILS [41]. It is a simple metaheuristic that is capable of dealing with a variety of optimization problems and that has been used successfully in many routing applications [42]. The algorithm explores the search space defined by a set of operators and returns solutions by iteratively refining the local optima. Through perturbations and local search phases, it escapes from local optima and enhances solution quality. Since we are dealing with a stochastic problem, before each acceptance, the new solution is evaluated by a recourse function, guiding the search towards the stochastic realization of the variables. The scenario-based iterated local search SBILS is shown in Algorithm 6. The algorithm runs until either a maximum run time or a maximum number of iterations without improvement is reached. Each step of the SBILS is described in detail in the following.

Algorithm 6 SBILS algorithm

procedure SBILS($T_{\max}$ = max run time, $I_{\max}$ = max number of iterations without improvements, p = perturbation intensity)
  $s^* \leftarrow$ CONSTRUCTIVEHEURISTIC
  $s^* \leftarrow$ VND(s^*)
  $Es^* \leftarrow$ EVALSCENARIOS(s^*) ▷ apply the recourse function in scenarios
  while ELAPSEDTIME $\leq T_{\max}$ and ITERATION $\leq I_{\max}$ **do**
    $s' \leftarrow$ PERTURBATION(s^*, p)
    $s'' \leftarrow$ VND(s')
    $Es'' \leftarrow$ EVALSCENARIOS(s'') ▷ apply the recourse function in scenarios
    if ACCEPT(Es^*, Es'') **then** ▷ the best of s^* and s'' with respect to $\overline{\mathcal{F}}$ (4.1)
      $s^* \leftarrow s''$
      $Es^* \leftarrow Es''$
      ITERATION $\leftarrow 0$
    else
      ITERATION $\leftarrow$ ITERATION $+ 1$
    end if
  end while
  return the best feasible solution found during the search
end procedure

Constructive heuristic. The initial solution is built by the constructive heuristic depicted in Algorithm 7. With the vertices of each period sorted in non-decreasing order of the start time of their time windows, the algorithm builds routes for each

period iteratively in two phases. First, the mandatory services are inserted sequentially in the end of each route. The second phase consists in the insertion of the optional services following the first fit approach. With this method, the route is iterated over and for each position the vertex is inserted, providing it does not generate any infeasibility. If the end of the route is reached without a successful insertion of the service, it is left unrooted.

Algorithm 7 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 ← FirstFit(σ_d, v)     ▷ insert optional v at the first feasible position
11:         end for
12:         end for
13:         return s = {σ_1, ..., σ_D}
14:     end procedure
```

Local search procedure. The local search procedure is conducted using a Variable Neighborhood Descent (VND) algorithm [20], which is crucial for the convergence of the search to a locally optimal solution. The combination of Iterated Local Search (ILS) and VND has been successfully applied in several optimization problems [61]. We are using the standard VND, called Basic Variable Neighborhood Descent (BVND) in [31], where the neighborhoods are used sequentially. The variant RVND, in which neighborhoods are chosen randomly, has shown to produce good results in many routing problems [62], but experiments have shown that in our problem it did not work well, as some neighborhoods are useful only when the simplest ones, such as removing optional elements, are applied first. The VND relies on a set of neighborhood structures N_k ($k = 1, \ldots, k_{\max}$) that are iteratively employed to explore the search space. Starting from neighborhood $k = 1$, the algorithm performs a local search using the Hill Climbing algorithm, ensuring the identification of the locally optimal solution within the current neighborhood k. Subsequently, the solution is evaluated to determine if it represents an improvement over the current best solution. If the solution is an improvement, the VND returns to the first neighborhood ($k = 1$) regardless of the current value of k. If not, it proceeds to the next neighborhood structure. The algorithm terminates when the last neighborhood structure is reached and no further improvements are found.

We implemented six neighborhood structures. Some of these are classical neighborhoods from the vehicle routing literature, while others were specifically designed to address the unique requirements of our problem. The neighborhoods are as follows:

- N_1 = Remove optional: Remove an optional service vertex from a route, thus trying to decrease the working time;
- N_2 = Swap: Swap the positions of two vertices inside a route;
- N_3 = 2-opt: Swap two arcs in a route, reversing the visiting order between the two arcs;

- N_4 = Relocate: Move a vertex to another position in the route;
- N_5 = Insert optional: Insert an optional unvisited vertex into a route, in an attempt to increase the collected score;
- N_6 = Swap optional: Swap two optional vertices, by letting an unvisited vertex take the place of a visited one.

The neighborhoods Swap, 2-opt, and Relocate can enhance the objective function solely by reducing the working time of a route, as they do not alter the collected score. The Remove optional movement seeks to decrease the working time, albeit at the cost of a reduced score. Conversely, the Insert optional neighborhood aims to improve the score, albeit with an increase in the working time. The final neighborhood, Swap optional, has the potential to enhance the objective function by either increasing the score, reducing the working time, or both.

It is important to note that all described neighborhoods consist of intra-period and intra-route movements, as they independently alter the routes within each period. A solution may be further enhanced by performing inter-period movements, which involve shifting the execution of a service at a customer from one period to another. This type of movement can reduce the working time in a period and create space for additional services to be performed. However, inter-period movements are costly to evaluate due to the large number of neighbors. Therefore, they are not fully explored in a deterministic manner 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 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.

Scenarios assessment. A key component in the proposed method is the solution assessment based on the scenarios' realization, done by the EVALSCENARIOS function depicted in Algorithm 8. This enables the solution search to be directed towards a resilient operational performance considering the scenario's probability. The assessment is done by applying in a deterministically generated solution each scenario $\omega \in \Omega$ by using a recourse function that mimics how a patrol should behave to respond to alarms.

The recourse function operates as follows. Upon receiving an alarm, it is necessary to determine its insertion point within the patrol route. Specifically, the alarm is incorporated between two consecutive visits such that the time interval between

Algorithm 8 Scenario evaluation algorithm

```
1: procedure EVALSCENARIOS(S)
2:     OF ← 0
3:     p ← 0
4:     for ω ∈ Ω do
5:         OF ← OF + ( P(ω)∗ RECOURSE(s, ω))   ▷ accumulate scenarios OF weighted by scenario probability
6:     end for
7:     return OF
8: end procedure
```

them encompasses the alarm's triggering time. Once this insertion point is identified, the patrol is immediately redirected to the alarm location.

Given the complexity of real-world scenarios, calculating the exact distance between the patrol's current position (while en route) and the alarm location is non-trivial. To address this, we consider two scenarios. The patrol can either (1) return to the location of the preceding visit and then proceed to the alarm location, or (2) continue to the location of the next scheduled visit, bypassing its service, and directly head to the alarm location. The recourse function evaluates these two options and selects the one with the minimum travel time. The alarm is then positioned within the route accordingly.

Upon addressing the alarm, the patrol resumes its routine operations. At this stage, the vehicle continues to the next scheduled service in a manner that minimizes disruptions to the remaining planned route. To achieve this, it may be necessary to skip certain planned services. If an optional service is skipped, its associated score is deducted from the total accumulated score. In contrast, skipping a mandatory service incurs an additional penalty on the loss of its score, weighted by a factor ϵ.

Figure 4.4 illustrates the operation of the recourse function under these conditions. Upon the triggering of alarm $a1$, the patrol considers two potential courses of action: (1) returning to service location 2, addressing alarm $a1$, and then resuming the planned itinerary, or (2) proceeding directly to service location 5, bypassing it, and subsequently addressing alarm $a1$. In the example shown, the first option is selected. Following this decision, the patrol evaluates how to continue with the remaining route. It may be the case that service 5 cannot be performed after addressing the alarm due to time windows. In that case, it is omitted (illustrated in the bottom route).

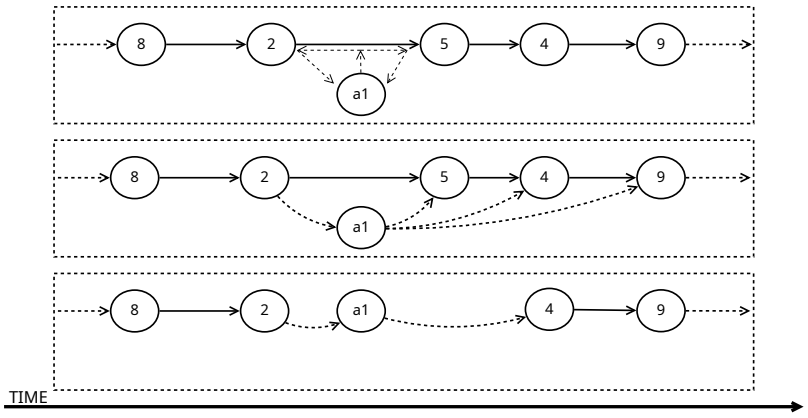


Figure 4.4: Recourse function with the addition of one alarm

Scenario generation. In our problem, scenarios are defined as subsets of alarms that are predicted in advance [1]. Each alarm includes three pieces of information: the location, determined by the customer's position; the trigger time; and the probability that the alarm is actually trigged. We define a scenario as a subset of all possible combinations of predicted alarms, with the probability of each scenario given by the product of the probabilities of its alarms. This results in a total of 2^n scenarios, where n represents the total number of alarm predictions. Therefore, a scenario reduction procedure is performed to manage the number of scenarios used. The objective of this procedure is to generate a representative subset of scenarios through scenario enumeration utilizing a binary tree structure. The process incorporates two backtracking criteria to limit the enumeration.

The first criterion is based on the current scenario's probability, which is incrementally calculated as the scenario is constructed. If the probability falls below a predefined threshold, further enumeration along that branch is terminated. The second criterion ensures diversity within the scenario pool. Each newly generated scenario is compared to the existing pool using the hamming distance and is added only if its distance from all previously included scenarios exceeds a specified threshold.

The enumeration process continues until the scenario sample reaches a predetermined size. The computational complexity of this procedure is influenced by the thresholds for backtracking, as they determine the extent of exploration required within the scenario tree. Lower probability thresholds and higher hamming distance thresholds can significantly increase the time needed to generate even a small sample, necessitating careful selection of these parameters.

4.6 Computational Results

In this section, we present the outcome of an extensive computational test performed on a large set of real-world instances provided by the company. The algorithms have been coded in c++20 and executed on a single thread. The MILP model was solved with Gurobi 11.0 and was invoked with its default configuration, letting it run on 12 threads. The experiments have been executed on an Intel Xeon CPU E5-2640 v3 2.60 GHz machine with 64 GB of memory, running under Windows 10 Pro 20H2 64-bits. We have run the algorithm five times and the average is reported. For what concerns the termination conditions, the ILS was allowed to run for at most $T_{\max} = 3600$ seconds on each province. On the basis of preliminary computational experiments, the value of parameter $I_{\max}$ has been set to 100.

4.6.1 Instances and scenario generation

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 4.1 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 number of predicted alarms ($|H|$), 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 4.1: Details of the real-world instances

Province	#	$\|D\|$	$\|H\|$	$tot_{\|C\|}$	$avg_{\|C\|}$	$max_{\|C\|}$	$min_{\|C\|}$	tot_n	avg_n	max_n	min_n
Mantova	2	7	16	43	21.5	32	11	171	85.5	140	31
Rimini	4	7	39	154	38.5	57	27	987	246.8	415	93
Sassari	7	7	16	119	17	33	7	1038	148.3	256	42
Roma	8	7	68	118	14.8	25	5	1234	154.3	268	84
Pescara	4	7	16	227	56.8	69	43	1370	342.5	484	129
Ravenna	5	7	28	172	34.4	50	15	1382	276.4	400	79
Ferrara	7	7	24	236	33.7	63	11	1812	258.9	525	97
Forli	8	7	26	407	50.9	73	16	2695	336.9	586	40
Bologna	8	7	161	239	29.9	63	17	2733	341.6	632	244
Parma	5	7	39	289	57.8	77	37	2952	590.4	797	323
Modena	11	7	70	510	46.4	76	9	4288	389.8	651	14
Reggio Emilia	10	7	123	679	67.9	100	22	7812	781.2	1280	243

The parameters α and β, utilized in the objective function, were determined by the company following internal discussions and set to 5 and 0.9, respectively. The parameter γ, responsible for the penalizing canceled mandatory service, was assigned a value of 10. The score obtained during a visit $\tau = 1, \ldots, n_{cdt}$ to customer c for service t within a given period was defined as $w_{ct\tau} = w_{ct}e^{1-\tau}$, where w_{ct} takes values from the set $\{1, 6, 9, 10\}$. The minimum quality of service (QoS) level was established at 75%, and the minimum time interval between two consecutive visits for the same service at a customer was set to $\delta_{\min} = 90$ minutes. Travel times were computed based on real-world distances using the Open Source Routing Machine (OSRM) application.

For scenario generation, a function based on the instance input was defined to establish a Hamming distance, which serves as one of the backtracking criteria. The function is expressed as:

$$D(H) = b + |H| \cdot p$$

where H represents the set of alarms, b is a base value ensuring a minimum Hamming distance regardless of the size of H, and p is a proportionality factor applied to $|H|$ to compute the Hamming distance. This formulation allows control over the diversity of scenarios generated by the procedure. A higher value of b promotes greater diversity in the scenario sample, whereas a lower value results in reduced diversity.

Based on empirical tests with the instances provided, b was set to 2, 4, and 8, while p was fixed at 0.025. The scenario sample sizes were defined as 30, 40, and 50. We report ahead the results for all combinations of b and sample size were tested, and the results are reported accordingly.

4.6.2 Mathematical model results

The mathematical model exhibited poor performance when applied to real-world instances, successfully solving only the Mantova instance, the smallest in the dataset. For the remaining instances, the solver ran out of memory while attempting to allocate the model. For the Mantova instance, the model achieved an objective value of 2522.37 with a GAP of 290% within a one-hour runtime. To verify the correctness of the solutions generated, a solution was produced for a small test instance specifically designed for validation purposes, as depicted in Figure 4.2. This test instance includes two customers, three services, and two alarms. Three scenarios are considered, covering all possible alarm configurations. The model generates a single route

for the deterministic scenario (where no alarms are present) and additional routes for each scenario to address the respective alarms.

4.6.3 SBILS results

Table 4.2 presents the percentage gap between the objective function values derived from the deterministic version of the algorithm and the corresponding values of these solutions when evaluated using the recourse function, applied post-optimization. This gap, calculated as:

$$\frac{\text{OF}_{\text{recoursed}} - \text{OF}_{\text{deterministic}}}{\text{OF}_{\text{recoursed}}} \times 100, \tag{4.25}$$

quantifies the relative loss in objective function value incurred when solutions derived from the deterministic version of the algorithm are evaluated under the recourse function. This indicates that failing to account for stochasticity in operational planning could result in significant performance losses as measured by the objective function. The analysis was conducted for b values set to 2, 4, and 8, and scenario sample sizes set to 30, 40, and 50. The consistently negative values across all tested configurations underscore the inadequacy of the deterministic algorithm in meeting the company's requirements, emphasizing that neglecting the stochastic nature of alarm occurrences would significantly impair operational effectiveness.

Table 4.2: Gap between the OF value obtained by a deterministically generated solution against the same solution after being submitted to the recourse function

Sample size	30			40			50		
b	2	4	8	2	4	8	2	4	8
Mantova	-15.2%	-15.2%	-15.2%	-15.3%	-15.3%	-15.3%	-16.0%	-16.0%	-16.0%
Rimini	-3.4%	-3.7%	-3.8%	-3.3%	-3.6%	-3.7%	-4.0%	-4.1%	-3.9%
Sassari	-8.4%	-7.2%	-7.7%	-8.3%	-7.2%	-7.6%	-8.2%	-9.8%	-7.2%
Roma	-4.9%	-12.5%	-12.0%	-9.6%	-12.0%	-9.8%	-12.7%	-13.1%	-14.4%
Pescara	-1.7%	-1.8%	-1.8%	-1.8%	-1.8%	-1.8%	-2.0%	-1.9%	-2.0%
Ravenna	-4.7%	-5.7%	-19.8%	-9.5%	-4.8%	-7.2%	-7.2%	-6.2%	-12.3%
Ferrara	-1.0%	-1.1%	-1.0%	-1.1%	-1.0%	-1.1%	-1.1%	-1.2%	-1.0%
Forli	-0.7%	-0.7%	-0.7%	-0.8%	-0.8%	-1.1%	-0.8%	-0.9%	-0.7%
Bologna	-0.5%	-0.5%	-0.4%	-0.4%	-0.4%	-0.6%	-0.6%	-0.5%	-0.9%
Parma	-0.8%	-0.7%	-0.7%	-0.7%	-0.5%	-0.6%	-0.7%	-0.9%	-1.1%
Modena	-1.7%	-1.7%	-1.7%	-1.7%	-1.7%	-1.7%	-1.8%	-1.7%	-1.7%
Reggio Emilia	-1.8%	-1.7%	-1.6%	-1.8%	-2.4%	-1.8%	-1.7%	-1.8%	-1.9%
Average	-3.7%	-4.4%	-5.5%	-4.5%	-4.3%	-4.4%	-4.7%	-4.8%	-5.3%

Table 4.3 presents the average results using all the combinations of scenarios parameters with deterministic and stochastic approaches. The column labeled $\bar{z}$ reports the objective function values for each approach. In the deterministic case, the value is obtained after optimization, with the recourse function applied to evaluate the solution generated deterministically across scenarios. The columns *RT* and *MCS* provide the average alarm response time and the average number of mandatory canceled services, respectively. The column *time* indicates the average computational runtime in seconds.

The average objective value of the solutions obtained through the stochastic approach is approximately 1.1% higher than that of the deterministic approach when evaluated across scenario realizations. Additionally, the stochastic approach resulted in a reduction in the average alarm response time, despite this metric not

Table 4.3: Results obtained by the deterministic and stochastic algorithm

	deterministic				stochastic			
Instance	$\bar{z}$	*RT*	*MCS*	*time*	$\bar{z}$	*RT*	*MCS*	*time*
Mantova	3661.2	3.6	0.0	4.0	3661.2	3.6	0.0	5.4
Rimini	19272.7	13.2	0.0	449.2	19359.7	13.4	0.0	527.7
Sassari	18481.6	75.1	0.0	90.8	18621.2	74.1	0.0	105.0
Roma	19869.0	17.6	0.0	131.3	20082.9	16.6	0.0	155.3
Pescara	36496.4	16.4	0.0	879.1	36656.7	16.9	0.0	1144.1
Ravenna	30239.0	10.4	0.9	1214.3	31551.8	10.3	0.2	1484.3
Ferrara	49170.9	21.4	0.0	1249.2	48979.2	19.1	0.0	1534.2
Forli	71986.7	7.8	0.0	2829.2	72167.7	7.6	0.0	3324.2
Bologna	62555.7	15.5	0.0	2132.6	64416.6	16.5	0.0	2506.6
Parma	76469.1	10.4	0.0	3600.0	76812.6	10.1	0.0	3600.0
Modena	96140.0	17.4	2.0	3600.0	96851.9	17.4	2.0	3300.0
Reggio	161556.7	18.1	0.5	3600.0	164077.4	18.1	0.3	3600.1
Average	53824.9	18.9	0.3	1648.3	54436.6	18.6	0.2	1773.9

being explicitly included in the objective function. Furthermore, the stochastic approach achieved an improvement in the average number of mandatory canceled services. Although this value was already low in the deterministic approach, the stochastic method reduced it even further. Furthermore, the solution time for the stochastic approach is only slightly higher.

4.6.4 Scenarios variance

The proposed algorithm comprises multiple components, with scenario generation playing a pivotal role in accurately representing the realizations of stochastic variables. To ensure effectiveness, we extensively tested two critical aspects of the scenario generation process: the Hamming distance, which facilitates backtracking, and the scenario sample size.

Table 4.4 presents the gap, as defined by equation (4.25) with $OF_{stochastic}$ replacing $OF_{recoursed}$, between the objective function values derived from the stochastic and deterministic solutions for each pairwise combination of scenario sample size and b values. This gap quantifies the improvement achieved by incorporating stochastic elements into the solution, using values of b set to 2, 4, and 8, and sample sizes of 30, 40, and 50. The results reveal several important trends. First, the average gaps across different sample sizes tend to be similar, with sample size 50 yielding slightly better average improvements. However, the configuration with sample size 30 and $b = 8$ demonstrates the most significant improvement, achieving an average improvement of 1.81%. Sample size 40 shows stable improvement across all values of b, suggesting robustness in performance. Overall, larger values of b, particularly $b = 8$, lead to greater performance gains, potentially reflecting their ability to capture more nuanced stochastic effects.

Analyzing the results by location, a few patterns stand out. For example, Ravenna exhibits a particularly large improvement of 11.87% under $b = 8$ and sample size 30, a result that heavily influences the average for this configuration. On the other hand, Ferrara consistently shows negative gaps across all parameter settings, indicating a potential limitation in the ability of stochastic solutions to outperform deterministic

ones in certain operational contexts. These location-specific variations may reflect differences in the characteristics of the input data, such as alarm distribution patterns or the geographical layout, which could merit further investigation.

Overall, the results emphasize that SBILS does not require a substantial number of scenarios to perform well, while greater scenario diversity tends to yield the most significant improvements. Also, these findings reinforce the importance of incorporating stochastic considerations into the solution process, as doing so ensures more robust and effective decision-making. Failure to adopt such approaches could lead to suboptimal outcomes and reduced operational effectiveness in practical contexts.

Table 4.4: Improvement between the stochastically generated solution and its deterministic counterpart

Sample size	30			40			50		
b	2	4	8	2	4	8	2	4	8
Mantova	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Rimini	-0.29%	0.15%	1.05%	0.53%	0.44%	0.46%	0.61%	0.80%	0.29%
Sassari	0.46%	0.75%	0.40%	1.34%	0.01%	0.84%	0.83%	2.37%	-0.26%
Roma	-6.05%	1.46%	0.85%	0.37%	1.33%	0.39%	1.35%	6.54%	3.08%
Pescara	0.46%	0.44%	0.44%	0.46%	0.38%	0.44%	0.46%	0.42%	0.43%
Ravenna	0.21%	1.49%	11.87%	6.15%	1.64%	3.91%	3.26%	1.00%	7.96%
Ferrara	-0.35%	0.18%	-0.30%	-0.50%	-0.45%	-0.46%	-0.54%	-0.40%	-0.71%
Forli	0.13%	-0.02%	0.09%	0.51%	0.17%	0.48%	0.42%	0.34%	0.12%
Bologna	2.98%	4.26%	4.21%	2.98%	3.19%	2.90%	1.67%	3.01%	0.74%
Parma	0.82%	0.33%	0.32%	0.26%	0.35%	0.33%	0.14%	0.68%	0.80%
Modena	0.42%	1.04%	0.92%	1.34%	0.27%	0.10%	0.87%	0.47%	1.19%
Reggio Emilia	1.38%	0.40%	1.91%	0.32%	2.09%	2.00%	1.69%	2.06%	1.93%
Average	0.01%	0.87%	1.81%	1.15%	0.79%	0.95%	0.90%	1.44%	1.30%

4.7 Conclusion

This study addresses a real-world car patrolling application encountered by a large Italian service company. The problem involves planning routes for a fleet of patrols that must deliver various security services to private customers. Each patrol is responsible for a cluster of customers, providing services on a weekly basis. Not all services can be performed, but a minimum service level must be maintained to ensure that a specified percentage of services is fulfilled. Additionally, some services, known as alarms, are not predetermined and may occur with a certain probability. The resulting optimization problem represents a complex variant of the team orienteering problem, incorporating additional constraints, particularly those related to uncertainty in service requests, along with stochastic service demands. We propose a two-stage stochastic programming model and a scenario-based ILS to address the problem.

Since this is a NP-hard problem, and we address large-scale real-world instances, the primary function of the model is to provide a mathematical representation that captures all components of the problem. The model was implemented using Gurobi but proved ineffective for solving the problem. Only the smallest instances could be solved, while for the remaining cases, the solver exhausted available memory.

The proposed metaheuristic incorporates a recourse function designed to guide the search process toward enhancing the realization of stochastic variables. The algorithm achieved a 1.1% improvement in solution quality compared to the deterministic version. Additionally, two key performance indicators (alarm response time and the cancellation rate of mandatory services) showed notable improvements.

For future research, we highlight two promising directions. First, the incorporation of parallelism offers significant potential to enhance scenario evaluation. While the recourse function did not substantially impact the algorithm's performance in the current work, we hypothesize that leveraging parallelism could enable the stochastic version of the algorithm to surpass the deterministic version in terms of performance. Second, an extension of this research to encompass online optimization represents another promising direction. The routes generated by our algorithm exhibit robustness, making them well-suited for managing real-world occurrences of uncertain events. Nevertheless, real-time adjustments, such as rerouting vehicles, remain essential to maintain operational continuity with minimal disruption when such events arise.

Chapter 5

Conclusion

This thesis addresses a complex, real-world Car Patrolling Problem (CPP), a critical challenge faced by a major Italian service provider. Through systematic study and dissemination in journals and conferences, the research contributes to multiple aspects of the problem. Chapter 2 introduces an Iterated Local Search (ILS) framework to tackle the deterministic routing component. Chapter 3 takes a strategic approach to territory division, proposing a two-stage algorithm to optimize patrolling areas. Finally, Chapter 4 focuses on the stochastic routing aspect, incorporating uncertainty to better reflect real operational conditions. Building upon these contributions, this thesis has demonstrated how tailored optimization approaches can address the unique challenges of the CPP while improving both operational efficiency and strategic planning.

The contributions of this thesis extend beyond the domain of the CPP. Problems with similar features can adopt our solution methods, with potential applications in fields such as tourism planning and goods delivery, further enhancing the impact of this work. While the thesis makes significant progress, it is important to acknowledge its limitations. The assumptions made to simplify stochastic routing and the challenges in scenario generation leave room for improvement. Future studies could explore the integration of online stochastic approaches to dynamically reroute during operation in response to alarm triggers. Advanced scenario generation techniques also represent a promising direction for further research.

Overall, this research shows how combining different methods can help solve real-world problems. By applying optimization techniques and practical insights, it tackles important challenges in patrolling and beyond. This work contributes to both the academic understanding of the problem and practical solutions for improving operations. Addressing the CPP is an important task, and this thesis aims to inspire further progress in this area.

Appendices

Appendix A

List of Acronyms

A.1 Acronyms, definitions, pages

CPP Car Patrolling Problem. 39

GRASP Greedy Randomized Adaptive Search Procedure. 26

ILS Iterated Local Search. iii, 7

KPI Key Performance Indicators. 16

MILP Mixed Integer Linear Programming. iii, 7, 42

MPOPTW Multi-Period Orienteering Problem. 10

MPTOPTWSD Multi-Period Team Orienteering Problem with Time Windows and Stochastic Demands. 44

OP Orienteering Problem. 6

QoS Quality of Service. 9

SBILS Simulation-Based Iterated Local Search. 49

TDMPTOPTW Territory-Design and Multi-Oeriod Team Orienteering Problem with Wime Windows. 28

TOP Team Orienteering Problem. 9

VND Variable Neighborhood Descent. 7

VNS Variable Neighborhood Search. 26

VRP Vehicle Routing Problem. 41

Bibliography

[1] K. Abreu et al. "Explainable Machine Learning for Alarm Prediction". In: *Proceedings of the 26th International Conference on Enterprise Information Systems - Volume 1: ICEIS*. INSTICC. SciTePress, 2024, pp. 690–697. ISBN: 978-989-758-692-7.

[2] M. Ahmed, R. Seraj, and S. M. S. Islam. "The k-means algorithm: A comprehensive survey and performance evaluation". In: *Electronics* 9.8 (2020), p. 1295.

[3] E. Angelelli et al. "The probabilistic orienteering problem". In: *Computers & Operations Research* 81 (2017), pp. 269–281.

[4] E. Angelelli et al. "A dynamic and probabilistic orienteering problem". In: *Computers & Operations Research* 136 (2021), p. 105454.

[5] C. Archetti, F. Carrabs, and R. Cerulli. "The set orienteering problem". In: *European Journal of Operational Research* 267.1 (2018), pp. 264–272.

[6] C. Archetti, M. G. Speranza, and D. Vigo. "Vehicle routing problems with profits". In: *in: P. Toth and D. Vigo, eds., Vehicle Routing: Problems, Methods, and Applications, second edition*. Milano: SIAM, 2014, pp. 273–297.

[7] H. Bakker, F. Dunke, and S. Nickel. "A structuring review on multi-stage optimization under uncertainty: Aligning concepts from theory and practice". In: *Omega* 96 (2020), p. 102080.

[8] E. Berhan et al. "Stochastic vehicle routing problem: A literature survey". In: *Journal of Information & Knowledge Management* 13.03 (2014), p. 1450022.

[9] M. Bernardo, B. Du, and J. Pannek. "A simulation-based solution approach for the robust capacitated vehicle routing problem with uncertain demands". In: *Transportation Letters* 13.9 (2021), pp. 664–673.

[10] J. R. Birge and F. Louveaux. *Introduction to stochastic programming*. New York: Springer, 2011.

[11] S. Biswas et al. "Geospatial clustering for balanced and proximal schools". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 09. 2020, pp. 13358–13365.

[12] P. S. Bradley, K. P. Bennett, and A. Demiriz. "Constrained k-means clustering". In: *Microsoft Research, Redmond* 20 (2000).

[13] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse. "The vehicle routing problem: State of the art classification and review". In: *Computers & industrial engineering* 99 (2016), pp. 300–313.

[14] A. M. Campbell, M. Gendreau, and B. W. Thomas. "The orienteering problem with stochastic travel and service times". In: *Annals of Operations Research* 186.1 (2011), pp. 61–81.

[15] J. G. Carlsson and E. Delage. "Robust partitioning for stochastic multivehicle routing". In: *Operations research* 61.3 (2013), pp. 727–744.

[16] V. H. V. Corrêa et al. *A scenario-based metaheuristic for the multi-period team orienteering problem with time windows and stochastic demands*. Tech. rep. University of Modena and Reggio Emilia, 2024.

[17] V. H. V. Corrêa et al. "Optimizing a Car Patrolling Application by Iterated Local Search". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '24. Melbourne, VIC, Australia: Association for Computing Machinery, 2024, 1201–1209. ISBN: 9798400704949.

[18] G. B. Dantzig and J. H. Ramser. "The truck dispatching problem". In: *Management science* 6.1 (1959), pp. 80–91.

[19] K. F. Doerner et al. "Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows". In: *Computers & Operations Research* 35.9 (2008), pp. 3034–3048.

[20] A. Duarte et al. "Variable neighborhood descent". In: *Handbook of heuristics* (2018), pp. 341–367.

[21] L. Evers et al. "A two-stage approach to the orienteering problem with stochastic weights". In: *Computers & Operations Research* 43 (2014), pp. 248–260.

[22] Forbes. *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]. 2017.

[23] D. Gavalas et al. "A survey on algorithmic approaches for solving tourist trip design problems". In: *Journal of Heuristics* 20.3 (2014), pp. 291–328.

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

[25] B. L. Golden, L. Levy, and R. Vohra. "The orienteering problem". In: *Naval Research Logistics (NRL)* 34.3 (1987), pp. 307–318.

[26] A. Gunawan, H. C. Lau, and K. Lu. "An iterated local search algorithm for solving the orienteering problem with time windows". In: *Evolutionary Computation in Combinatorial Optimization: 15th European Conference, EvoCOP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings 15*. Springer. 2015, pp. 61–73.

[27] A. Gunawan, H. C. Lau, and K. Lu. "SAILS : Hybrid Algorithm for the Team Orienteering Problem with Time Windows". In: *Proceedings of the 7th multidisciplinary international scheduling conference* (2015), pp. 276–295.

[28] A. Gunawan, H. C. Lau, and K. Lu. *Well-Tuned ILS for Extended Team Orienteering Problem with Time Windows*. Tech. rep. LARC-TR-01-15. Available at http://research.larc.smu.edu.sg/larcweb/larc/publications/technicalreports/Well-Tuned-ILS-for-Extended-Team-Orienteering-Problem-with-Time-WindowsTR-01-15.pdf: Living Analytics Research Center, 2015.

[29] A. Gunawan, H. C. Lau, and P. Vansteenwegen. "Orienteering problem: A survey of recent variants, solution approaches and applications". In: *European Journal of Operational Research* 255.2 (2016), pp. 315–332. ISSN: 03772217.

[30] F. Gündling and T. Witzel. "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. 2020.

[31] P. Hansen et al. "Variable neighborhood search: basics and variants". In: *EURO Journal on Computational Optimization* 5.3 (2017), pp. 423–454.

[32] P. Hansen et al. "Variable neighborhood search". In: *in: M.Gendrau and J.-Y. Potvin, eds., Handbook of Metaheuristics*. New York: Springer, 2019, pp. 57–97.

[33] T. Ilhan, S. M. Iravani, and M. S. Daskin. "The orienteering problem with stochastic profits". In: *Iie Transactions* 40.4 (2008), pp. 406–421.

[34] L. Jiao et al. "A multi-stage heuristic algorithm based on task grouping for vehicle routing problem with energy constraint in disasters". In: *Expert Systems with Applications* 212 (2023), p. 118740.

[35] A. A. Juan et al. "A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems". In: *Operations Research Perspectives* 2 (2015), pp. 62–72.

[36] O. Kariv and S. L. Hakimi. "An algorithmic approach to network location problems. I: The p-centers". In: *SIAM journal on applied mathematics* 37.3 (1979), pp. 513–538.

[37] M. Keskin, B. Çatay, and G. Laporte. "A simulation-based heuristic for the electric vehicle routing problem with time windows and stochastic waiting times at recharging stations". In: *Computers & Operations Research* 125 (2021), p. 105060.

[38] M Khodadadian et al. "Time dependent orienteering problem with time windows and service time dependent profits". In: *Computers and Operations Research* 143.March (2022), p. 105794. ISSN: 0305-0548.

[39] S. Kotiloglu et al. "Personalized multi-period tour recommendations". In: *Tourism Management* 62 (2017), pp. 76–88.

[40] I. Litvinchev, G Cedillo, and M Velarde. "Integrating territory design and routing problems". In: *Journal of Computer and Systems Sciences International* 56 (2017), pp. 969–974.

[41] H. R. Lourenço, O. C. Martin, and T. Stützle. "Iterated local search". In: *Handbook of metaheuristics*. Springer, 2003, pp. 320–353.

[42] H. R. Lourenço, O. C. Martin, and T. Stützle. "Iterated local search: Framework and applications". In: *in: M.Gendrau and J.-Y. Potvin, eds., Handbook of Metaheuristics*. New York: Springer, 2019, pp. 129–168.

[43] Y. Lu, U. Benlic, and Q. Wu. "A memetic algorithm for the orienteering problem with mandatory visits and exclusionary constraints". In: *European Journal of Operational Research* 268.1 (2018), pp. 54–69.

[44] J. G. Moya-García and M. A. Salazar-Aguilar. "Territory design for sales force sizing". In: *Optimal districting and territory design* (2020), pp. 191–206.

[45] J. Oyola, H. Arntzen, and D. L. Woodruff. "The stochastic vehicle routing problem, a literature review, part II: solution methods". In: *EURO Journal on Transportation and Logistics* 6.4 (2017), pp. 349–388.

[46] J. Oyola, H. Arntzen, and D. L. Woodruff. "The stochastic vehicle routing problem, a literature review, part I: models". In: *EURO Journal on Transportation and Logistics* 7.3 (2018), pp. 193–221.

[47] P. J. Palomo-Martínez et al. "A hybrid variable neighborhood search for the orienteering problem with mandatory visits and exclusionary constraints". In: *Computers & Operations Research* 78 (2017), pp. 408–419.

[48] J. Panadero et al. "A simheuristic approach for the stochastic team orienteering problem". In: *2017 Winter Simulation Conference (WSC)*. IEEE. 2017, pp. 3208–3217.

[49] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

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

[51] J. Reese. "Solution methods for the p-median problem: An annotated bibliography". In: *Networks* 48.3 (2006), pp. 125–142.

[52] R. Z. Ríos-Mercado. *Optimal districting and territory design*. Vol. 284. Springer, 2020.

[53] R. Z. Ríos-Mercado and J. C. Salazar-Acosta. "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. 2011, pp. 307–318.

[54] U. Ritzinger, J. Puchinger, and R. F. Hartl. "A survey on dynamic and stochastic vehicle routing problems". In: *International Journal of Production Research* 54.1 (2016), pp. 215–231.

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

[56] S. Samanta, G. Sen, and S. K. Ghosh. "A literature review on police patrolling problems". In: *Annals of Operations Research, 316* (2021), pp. 1063–1106.

[57] S. Samanta, G. Sen, and S. K. Ghosh. "Correction to: A literature review on police patrolling problems". In: *Annals of Operations Research, 316* (2022), p. 1575.

[58] M. Schneider et al. "Territory-based vehicle routing in the presence of time-window constraints". In: *Transportation Science* 49.4 (2015), pp. 732–751. ISSN: 15265447.

[59] E. Schubert et al. "DBSCAN revisited, revisited: why and how you should (still) use DBSCAN". In: *ACM Transactions on Database Systems (TODS)* 42.3 (2017), pp. 1–21.

[60] A. Shapiro et al. "Risk neutral and risk averse Stochastic Dual Dynamic Programming method". In: *European Journal of Operational Research* 224.2 (2013), pp. 375–391.

[61] A. Subramanian and L. dos Anjos Formiga Cabral. "An ILS based heuristic for the vehicle routing problem with simultaneous pickup and delivery and time limit". In: *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer. 2008, pp. 135–146.

[62] A. Subramanian and H. R. Lourenço. "Iterated Local Search". In: *Encyclopedia of Optimization*. Ed. by P. M. Pardalos and O. A. Prokopyev. Cham: Springer International Publishing, 2020, pp. 1–10. ISBN: 978-3-030-54621-2. DOI: 10.1007/978-3-030-54621-2_798-1. URL: https://doi.org/10.1007/978-3-030-54621-2_798-1.

[63] Q. Sun, H. Zhang, and J. Dang. "Two-stage vehicle routing optimization for logistics distribution based on HSA-HGBS algorithm". In: *IEEE Access* 10 (2022), pp. 99646–99660.

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

[65] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden. "The orienteering problem: A survey". In: *European Journal of Operational Research* 209.1 (2011), pp. 1–10.

[66] V. H. Vidigal Corrêa et al. "An iterated local search for a multi-period orienteering problem arising in a car patrolling application". In: *Networks* 83 (2024), pp. 153–168.

[67] A Villalba and E. Rotta. "Clustering and heuristics algorithm for the vehicle routing problem with time windows". In: *International Journal of Industrial Engineering Computations* 13.2 (2022), pp. 165–184.

[68] R. T. Wong. "Vehicle routing for small package delivery and pickup services". In: *The Vehicle Routing Problem: Latest Advances and New Challenges*. Boston, MA, USA: Springer, 2008, pp. 475–485.

[69] W. Xu et al. "Approximation algorithms for the generalized team orienteering problem and its applications". In: *IEEE/ACM Transactions on Networking* 29.1 (2021), pp. 176–189.

[70] F. Zehra et al. *Comparative analysis of C++ and python in terms of memory and time*. Tech. rep. 2020120516. Available at https://www.preprints.org/manuscript/202012.0516/v1, 2020.

[71] S. Zhang, J. W. Ohlmann, and B. W. Thomas. "Multi-period orienteering with uncertain adoption likelihood and waiting at customers". In: *European Journal of Operational Research* 282.1 (2020), pp. 288–303.

[72] L. Zhou et al. "A Heuristic Algorithm for solving a large-scale real-world territory design problem". In: *Omega* 103 (2021), p. 102442.

[73] G. Zucchi et al. "A Metaheuristic Algorithm for a Multi-period Orienteering Problem arising in a Car Patrolling Application". In: *Proc. 10th International Network Optimization Conference (INOC) Aachen, Germany, March 1-4*. 2022, pp. 99–104.