An Optimization-based Decision Support System for Multi-Trip Vehicle Routing Problems

Mirko Cavecchia[1*], Thiago Alves de Queiroz[2], Manuel Iori[1], Riccardo Lancellotti[3], Giorgio Zucchi[4]

[1]Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Via Giovanni Amendola 2, 42122, Reggio Emilia, Italy.
[2]Institute of Mathematics and Technology, Federal University of Catalão, 75704-020, Catalão-GO, Brazil.
[3]Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Via Pietro Vivarelli 10, 41125, Modena, Italy.
[4]R&D department, Coopservice S.coop.p.a, Via Rochdale 5, 42122, Reggio Emilia, Italy.

*Corresponding author(s). E-mail(s): mirko.cavecchia@unimore.it;
Contributing authors: taq@ufcat.edu.br; manuel.iori@unimore.it; riccardo.lancellotti@unimore.it; giorgio.zucchi@coopservice.it;

Abstract

Emerging trends, driven by industry 4.0 and Big Data, are pushing to combine optimization techniques with Decision Support Systems (DSS). The use of DSS can reduce the risk of uncertainty of the decision-maker regarding the economic feasibility of a project and the technical design. Designing a DSS can be very hard, due to the inherent complexity of these types of systems. Therefore, monolithic software architectures are not a viable solution. This paper describes the DSS developed for an Italian company based on a micro-services architecture. In particular, the services handle geo-referenced information to solve a multi-trip vehicle routing problem with time windows. To face the problem, we follow a two-step approach. First, we generate a set of routes solving a vehicle routing problem with time windows using a metaheuristic algorithm. Second, we calculate the interval in which each route can start and end, and then combine the routes together, with an integer linear programming model, to minimize the number of used vehicles. Computational tests are conducted on real and random instances and prove the efficiency of the approach.

Keywords: Decision Support System, Micro-Services, Multi-Trip Vehicle Routing Problem, Iterated Local Search

1 Introduction

The vehicle routing problem (VRP) is a challenging combinatorial optimization problem that has gained considerable attention from researchers, given its practical applications in different fields such as logistics and transportation. The main objective of the VRP is to find a set of routes for a fleet of vehicles that minimizes the total cost while serving the demand of a set of geographically dispersed customers. Due to its NP-hard nature, advanced heuristics and metaheuristics algorithms have been proposed to find approximate but good-quality solutions of large-size instances within reasonable computing times [1–3].

In this paper, we address a variant of the VRP known as the multi-trip vehicle routing problem with time windows (MTVRPTW), which arises from a real-world application at Coopservice Soc.coop.p.A, an Italian big service provider company that needs to serve customers by delivering products from its large depots. In this problem, the routes to be determined must respect time windows constraints, which in turn imposes each vehicle to serve a customer within specified time slots. Besides that, vehicles with different capacities can depart from different depots and each vehicle can perform more than one route. Each route must start and end at the same depot. In our case, the objective is to minimize the number of vehicles used to fulfill customers' demands.

To tackle the MTVRPTW, we propose a decision support system (DSS) designed to assist the company in its operations. The DSS is a web application composed of different micro-services. The micro-services architecture guarantees scalability with respect to the workload (new instances of the micro-services can be added to cope with traffic surges) and it ensures interoperability with other systems that can be easily interfaced with the DSS. The orchestration of the micro-services has a single entry point where the user can enter the addresses of depots and customers. The addresses are then geo-referenced and the distance between each couple of addresses is computed. Next, we generate a set of feasible routes using the metaheuristic approach defined in Kramer et al. [4]. However, since the vehicles can perform more than one route in a day, these routes need to be aggregated in order to minimize the fleet size.To this end, we consider two situations, in accordance with the company. In the first scenario, the starting time of each route is fixed, and we propose a greedy algorithm that sorts routes based on their departure time, assigning them to vehicles while respecting the total driving time. In the second scenario, the starting time of each route can be moved between the earliest and latest starting time. In this case, we extend the greedy algorithm to assign routes to vehicles as early as possible they can start; we propose an iterated local search (ILS) metaheuristic to improve the solution from the greedy algorithm by applying a perturbation step followed by a local search, based on swap and relocate movements. Additionally, we propose a mixed integer linear programming model to compare its solution with the other algorithms. These algorithms are evaluated on realistic instances provided by the industrial partner, with the ILS showing the best overall results.

To the best of our knowledge, the DSS represents a significant innovation for the reference case study used by the industrial partner. The micro-service based architecture is a state-of-the-art technology in the area of DSS, offering benefits in terms of scalability and interoperability. Furthermore, the use of multiple algorithms, which can be flexibly combined, guarantees the support for evaluating multiple solution approaches suitable for different scenarios. Finally, the open architecture based on micro-services simplifies the extension of the core framework in order to implement additional algorithms that can be suitable for other specific case studies.

A preliminary version of the present research has been published in Cavecchia et al. [5]. This paper is a clear step ahead providing more algorithms that have been added to the framework and outlining a broader performance evaluation based on several realistic data sets.

The remainder of this paper is structured as follows. Section 2 provides an overview of the existing literature in the fields of routing problems and decision science. Section 3 outlines the business process and the micro-services architecture developed to support the process. In Section 4 a formal description of the optimization problem is provided. Section 5 details the two-step approach proposed to solve the problem. Section 6 provides a performance evaluation of the DSS with respect to different case studies. Finally, Section 7 concludes the paper by providing final insights on the problem and its solutions.

2 Literature Review

This section provides an analysis of the existing literature concerning the VRP and its variants, with a specific emphasis on the MTVRPTW. Additionally, it explores the domain of DSSs and their relevance within the research context.

The VRP may consider additional characteristics and constraints, called attributes, resulting in multi-attributes VRPs. In their survey, Vidal et al. [6] reviewed sixty-four heuristics and metaheuristics algorithms, identifying the core components of each one and examining fifteen notable problems. In real-world scenarios, many companies and researchers have developed tailored optimization algorithms to solve specific VRP instances. For example, Mendes and Iori [7] addressed a combined problem involving VRPTW and scheduling of trucks and drivers. While these approaches provided effective solutions for their respective problems, they often did not fully consider the practical aspects and challenges that arise during real-world implementations.

When multiple attributes are combined together, we have the so-called rich VRPs. A recent survey for rich variants was proposed by Goel and Bansal [8], who presented a literature review on hybrid methods and problems found, e.g., in food and newspaper distributions. The authors emphasized the importance of cooperative methods that combine exact and approximate algorithms, as well as the proposal of benchmark problems including real-life instances. For a concise review of VRP and its variants, we refer to Vidal et al. [9], while Mor and Speranza [10] reviewed the variants with profits, split deliveries, multiple commodities over time, and integrated VRPs, like location-routing problems, multi-echelon routing problems, and the routing problems with loading constraints.

The VRPTW, an extension of the classical VRP, introduces temporal constraints on customer service. Each customer must be visited within a predefined time window. This additional constraint adds complexity to the problem, making it even more challenging to solve. Over time, starting from Desrochers et al. [11], researchers have proposed various exact and heuristic methods to tackle the VRPTW [12, 13]. The applications of the VRPTW span across various domains, including food delivery [14], electric vehicle recharging [15], and pharmaceutical product delivery [4].

Another well-established VRP variant is the MTVRPTW, where vehicles can undertake multiple routes in a single day, with each route starting and ending at the depot. This flexibility enables better adherence to customer time windows and can lead to more efficient solutions. The MTVRPTW has been an area of significant research interest, and various algorithms have been recently proposed to address it effectively [10, 16].

Many solutions and methods for the MTVRPTW can be found in the literature. Masmoudi et al. [17] solved a variant related to the dial-a-ride problem, proposing hybrid bee colonies algorithms and an adaptive large neighborhood search. The authors also proposed a new set of instances, comprising small, medium and large ones. Better results are obtained with the hybrid algorithms. Wang [18] solved a variant found in the meal delivery logistics, proposing an integer linear programming model and two heuristics, an iterated local search and an adaptive large neighborhood search. The results indicate the iterated local search is able to optimally solve small instances, while the other heuristic could improve the iterated local search results by 2% on average. Pan et al. [19] handled the multi-trip time-dependent vehicle routing problem with time windows by proposing a mixed integer programming model and a hybrid adaptive large neighborhood search for guided exploration and the variable neighborhood descend for intensive exploitation. The authors proposed new instances and their heuristics are competitive on these instances. Recently, Nguyen et al. [20] solved a multi-trip multi-distribution center vehicle routing problem with lower-bound capacity constraints, with the aim of minimizing the number of used vehicles and maximizing the number of served customers. To handle the lower-bound capacity constraints, the authors considered greedy algorithms based on savings, insertions, and sweeps. The solutions obtained with the greedy algorithms are then improved by an adaptive large neighborhood search.

With respect to the second contribution of the paper, the application area of decision support systems spans from business and finance to healthcare and up to education. Several criteria can be used to categorize this type of system such as their scope or the provided functionalities. A possible classification proposed in Power and Sharda [21] divides DSSs into communications-driven, data-driven, document-driven, knowledge-driven, and model-driven. The proposed approach in this paper fits a model-driven approach as the DSS focuses on capturing a business process that is described as the composition of multiple basic functions.

The literature on DSS presents a large corpus investigating whether micro-services or cloud-based architectures are viable options for the management of services. For example, Farshidi et al. [22] proposed a DSS that aims to evaluate if an application is suitable for cloud deployment. In a similar way, Christoforou et al. [23] focuses on micro-service-based deployments, proposing a tool to evaluate the benefits of architectural re-design following the micro-services paradigm. With respect to this topic, our study clearly embraces the vision of a highly dynamic description of the business process, where data are not a critical asset but rather a commodity. To this aim, we explicitly embrace a micro-service architecture for the deployment of the DSS itself. Indeed, the need of our application clearly fits the critical parameters that make micro-service appealing, as described by Di Francesco et al. [24].

Finally, the considered problem fits the area of Spatial-based DSS, which has been explored by Keenan and Jankowski [25]. The proposed system fits in the area of classic DSS with optimization based on geo-referenced data. However, even if the core problem is classic, the approach of creating an agile, flexible and scalable system takes advance of the most recent advances in distributed systems and cloud computing.

3 Decision Support System Architecture

We can define a decision support system as an information system based on computers that support the task of taking decisions. In this section we will outline the business process used in the proposed DSS and we will discuss how to implement such process.

3.1 Business process overview

In this paper we propose a model-driven DSS aiming to support a specific business process. To better explain the characteristics of the proposed system, we start with a short description of the main tasks that are required to solve MTVRPTW problems.

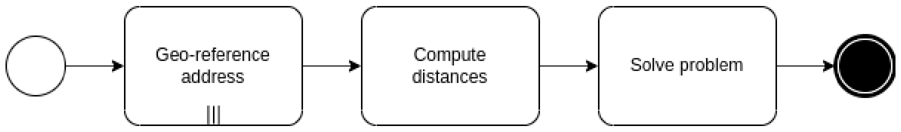

Fig. 1 BPM description of the business process

Figure 1 shows the main tasks of the considered business process according to a BPMN notation. The input is a list of trip way-points in the form of addresses that are geo-referenced (as shown in the leftmost box in the graph). The list of coordinates is further processed to create a distance matrix where each element of the matrix is the distance between two way points (the distance can be expressed both in terms of time or as a physical distance (i.e. km), depending on the metric considered for the problem). The matrix is then used as an input for the MTVRPTW, which is then solved using a two-step approach.

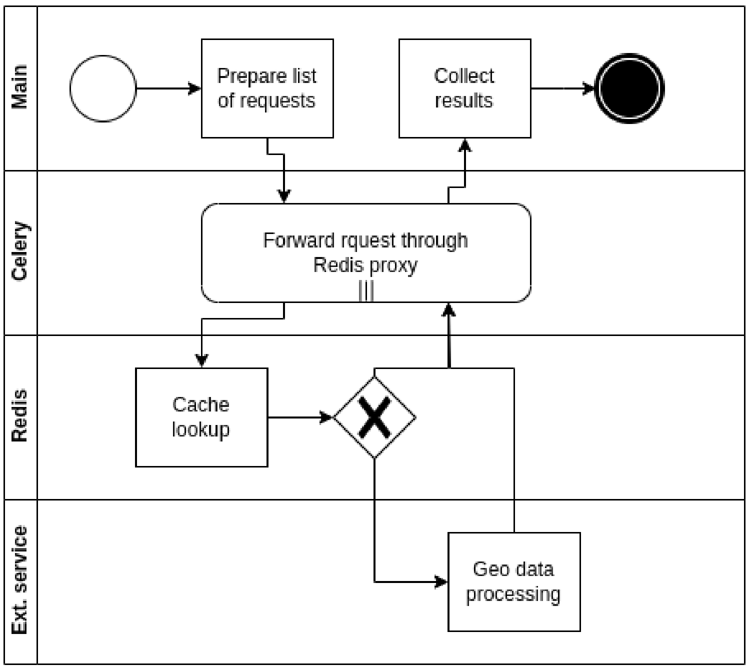

Fig. 2 Address geo-referencing and distance computation

The two tasks of geo-referencing and distance calculation require complex operations and rely on external services (such as APIs provided by third party microservices). As an example, ArcGIS provides an API with python bindings for most of its operations as documented in [26]. However, external services typically impose a limit on the number of invocations per unit of time.

This constraint requires a suitable architectural solution, as shown in Figure 2. Considering the tasks of geo-referencing the way-points or computing distances, the requests are inserted in a queue of requests that are handled by Celery[1]. Each celery Task ID identifies a list of requests. The Task ID can be used to track the state of a Task (i.e., how many requests have already been completed). A further element is inserted between Celery and the external APIs that is a cache implemented through Redis[2]. Resolutions of requests are stored by Redis and can be used to improve the scalability of the system both by reducing the time to interrogate external servers and by avoiding wait time due to the invocation rate limitations.

3.2 Services definition

Focusing on the services for the vehicle routing application, the services are described using the OpenAPI specification v2.0. For space reasons we provide only a short summary of the services.

[1] https://docs.celeryq.dev/
[2] https://redis.io/

Focusing on the first service, which is geo-referencing a list of addresses, can be described as the following set of micro-services:

- Submission of a list of addresses for geo-referencing.: for the sake of interoperability with other tasks of the company, the list of supported input formats include also `.xls` files. The output of the input is a handle that includes the ID of the geo-referencing task that runs asynchronously with respect to the end of the submission.
- Status of a task: the input is the handle with the task ID provided by the previous service and the output is the number of resolved addresses up to now. This service can be invoked by the UI to provide feedback to the user on the progress of a task.
- Download of coordinates lists: this service returns data only after the task is completed, otherwise an error code is returned. The data can be returned both as `JSON` and as an `.xls` file. The first type of output is used to display a map with the results of the geo-referencing (using a mash-application that is based on Open Street Maps APIs), while the `.xls` file is used as the input of the subsequent task of computing a distance matrix between every couple of points.

The API for the second task of the business process described in Figure 1 is similar to the previous task. The main difference is that the main input is a `.xls` file with the coordinates of the gro-referenced point previously obtained, while the output is another `.xls` file containing the distance matrix. Like the previous task, the execution is asynchronous with a submission micro-service that returns a handle used to check the progress of the computation.

Finally, the last task shown in Figure 1 is the resolution of the optimization problem. The solution is a two-step process, but the main API implementation masks this double step under two separate APIs:

- Problem submission: the API is used to submit a problem definition consisting in a distance matrix and an expected workload. The output is a handle to access the results of the algorithm invocation.
- Solution download: the handle provided in the problem submission API is used to access the data containing the problem solution. The output can be either a `JSON` data structure or a `.xls` file. The first is used for data visualization on a Web UI, while the latter can be downloaded for interaction with the other tasks of the company.

3.3 Technologies

As the goal is to provide support in logistic optimization tasks that must be carried out by people that are not experts in using computer systems, the proposed algorithms are integrated into an intuitive and user-friendly Web-based user interface. The UI leverages the modular architecture of the software that is provided as a suite of micro-services.

The micro-services approach to software development prescribes that software should be split into several independent building blocks that are loosely coupled. This approach made software development extremely agile, with each service choosing its own set of technologies, scalability policies and life cycle. This is a clear step ahead over monolithic architectures in terms of scalability, maintainability, and fault tolerance [27].

The global DSS architecture is composed of two main parts: a back-end and a front-end. The back-end handles HTTP requests, performs computations and store data, providing the previously described micro-services interfaces to the system. The back-end is developed using a model-view-controller paradigm [28] provided by the Web framework Django. The front-end runs on the user web browser, is written in Javascript and is based on the React framework to provide a simple interface for the end-user.

A key technology selected for the deployment of the DSS back-end is the container management engine Docker[3], which is used also in the testing and development phases of the micro-services. Each task and the related micro-services are implemented as a set of isolated images (containers) that can be created, replicated and destroyed using a simple set of command line tools.

The back-end deployment scheme consists of several containers outline by green boxes in Figure 3:

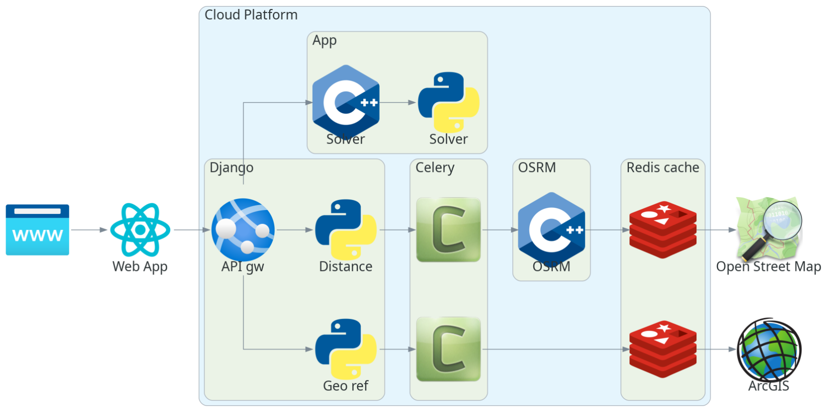

Fig. 3 Web Interface Architecture

- *App* contains the optimization algorithms, with the two parts of the solution algorithms implemented in C++ and Python
- *OSRM* is a C++ routing service designed to interrogate the Open Street Map APIs

[3]https://docs.docker.com/

- *Celery* is an asynchronous task queue manager used to automate multiple requests to different endpoints without the risk of overloading the external services
- *Redis Cache* is a service that manages the sending, receiving, and queuing of messages with Celery
- *Django* manages the API endpoints and provides the micro-services APIs.

The logical components of the software are also detailed in Figure 3. These components can be detailed as follows:

- Geo-reference: this module is responsible for the fist task of the BPMN model shown in Figure 1, that is to retrieve the coordinates of a list of addresses
- Travel matrix generation: this module is in charge of the second step in Figure 1, that is to create a distance and time matrix from the list of coordinates
- MTVRPTW solver: this module is in charge of the resolution of the MTVRP starting from customers' time slots, depots information, and vehicle types characteristics. The problem is detailed in Section 4, while the algorithm to tackle the problem is described in Section 5

As an example of the Web UI we also provide a screenshot in Figure 4. We observe a graphical representation of the computation output (in the form of a map with way-points and routes), together with a tabular representation of the solver output (with schedules and travel distances) that can be downloaded as an Excel file. Several buttons are used to upload the problem definition, start the solver execution, plot results or download them.

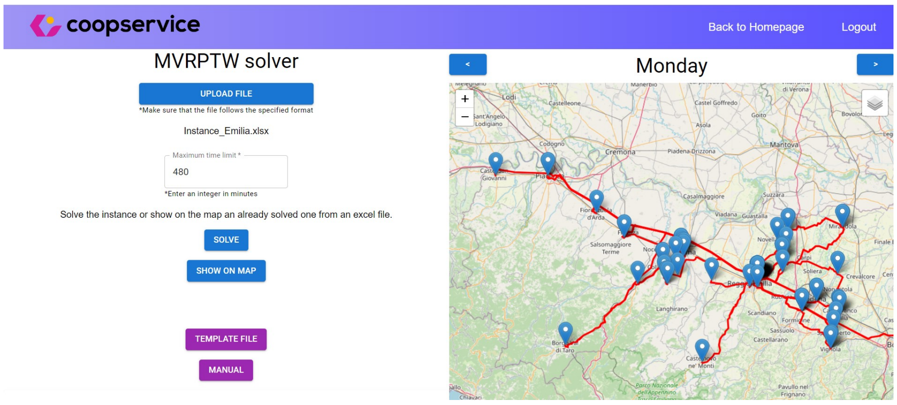

Fig. 4 A screenshot of the MTVRPTW module

4 Problem Description

We define the MTVRPTW on a directed graph $G = (N, A)$ with a set of nodes N and a set of arcs (i.e., directed edges) $A = \{(i, j) : i, j \in N, i \neq j\}$. The set of nodes N is categorized into depots (D) and customers (C). Each arc $(i, j) \in A$ is associated with a traveling time t_{ij}. Additionally, each node $i \in N$ is associated with a specific time window $[e_i, l_i]$, where e_i represents the earliest arrival time and l_i represents the latest arrival time. If a vehicle arrives before e_i, it must wait, and it cannot arrive after l_i.

Multiple-day deliveries may be required for each customer. Let P denote the set of days in which the planning is required. Each customer $i \in C$ is characterized by a demand q_{ip} on the day $p \in P$ and along with a service time s_i. The vehicle fleet, represented by the set V, is heterogeneous and is divided into different types. Vehicles of the same type are defined by K_v, and all vehicles $k \in K_v$ are identical in terms of loading capacity and permissible routes (e.g., mountainous arcs can be traveled only by the smallest vehicles).

A feasible solution for the problem must satisfy the following constraints: each route is associated with a unique depot and respects the vehicle's capacity; a vehicle can perform multiple routes in a day and each route needs to start and ends at the same depot; each customer is assigned to exactly one route and the total demand must be accomplished during a single visit within the time window. Additionally, the sum of the durations of the routes assigned to each vehicle cannot exceed $T = 480$ minutes per day. Furthermore, before starting another route, the vehicle requires a fixed loading time of $\Delta = 30$ minutes, which is included in the overall time limit T.

The objective of the problem is to find a set of routes that satisfy the aforementioned constraints while minimizing the number of used vehicles. All vehicles can operate on any day in set P. The problem is solved separately for each day $p \in P$, and the solution for one day does not depend on the solutions for other days.

5 Proposed Methodology

In this section, we present our two-phase decomposition algorithm. According to Santini et al. [29], this kind of strategy has allowed to obtain high quality results for VRPs and their variants. In the first phase of our algorithm, we solved an instance of the VRPTW. The objective here is to obtain a set R of routes satisfying the customers' demands and other operational constraints. The set R is then used as input for the second phase. The objective in this phase is to minimize the number of vehicles when solving an instance of the MTVRPTW. In other words, Section 5.1 presents the first phase, while Section 5.2 discusses in detail all the algorithms we have proposed, namely a greedy algorithm with fixed starting times, a greedy algorithm with bounded starting times, an iterated local search, and a mixed integer programming mathematical model.

5.1 Solving the VRPTW

To solve the VRPTW, we used the algorithm proposed by Kramer et al. [4]. They solved a real-world distribution case study in Coopservice, considering multiple depots, a heterogeneous fleet of vehicles, flexible time windows, periodic demands, incompatibilities between vehicles and customers, a maximum duration for the routes, and a maximum number of customers per route. The authors proposed a multi-start iterated local search algorithm making use of several neighborhood operators.

The first step in the multi-start iterated local search is to obtain an initial solution from a constructive heuristic. The constructive heuristic creates routes by adding customers to the closest depot by inserting customers one at a time in the route that generates the lowest cost. This heuristic allows time window violations, but they are penalized when calculating the objective function.

The next step is to improve the initial solution by means of local search. It consists of a randomized variable neighborhood descent and contains neighborhoods based on inter- and intra-route movements. Besides that, perturbation procedures are applied on the solution to escape from local optima. They consist of modifying the routes by using random swap movements and customer relocations.

After all, we use this algorithm to obtain a set R of routes that are used as input for the MTVRPTW. Since our goal is to find the minimum number of used vehicles, we aim to combine the routes of R. In other words, depending on the customers' time windows, a vehicle can return to the depot and perform another route, reducing the number of vehicles needed. Therefore, we propose four algorithms, explained in the next section, to attain this objective.

5.2 Solving the MTVRPTW

To solve the presented problem, we propose four algorithms: a greedy with fixed starting times, a greedy with bounded starting times, an iterated local search and a mathematical model. All the proposed methods take as input the set R of routes generated by the algorithm of Kramer et al. [4].

The first approach that we have used is to take the route generated and try to aggregate them into multiple duties to assign the aggregated routes to the vehicles. In order to do so we have developed a heuristic approach with fixed starting times. The details of the algorithm are presented in the next session.

Furthermore, since the time windows are not so restricted, instead of assuming that the starting time of each route is fixed, we accept to modify it by still ensuring that all customers in the route are visited within their time windows. In this case, the problem we face is to define the starting time of each route so as to minimize the number of used vehicles.

This problem has to be solved for each vehicle type $v \in V$, for each depot $d \in D$, and for each day $p \in P$. This is due to the fact, as already mentioned, that only routes that depart on the same day from the same depot and use the same vehicle type can be aggregated with the other.

To solve this problem we have decided to use the forward-time slack procedure proposed by Savelsbergh [30]. Let $r \in R$ be a route, and N_r be the sequence of nodes visited by r. For each node $i \in N_r$, we define ST_i as the earliest feasible start time, WT_i as the cumulative idle time, and FT_i as the partial forward slack time. To find the start time of route r, we initially set $ST_1 = e_1$, $WT_1 = 0$, and $FT_1 = l_1 - e_1$. For the next nodes $i \in N_r$, we calculate:

$$ST_i = \max(ST_{i-1} + t_{i-1,i} + s_i;\ e_i + s_i), \tag{1}$$

$$WT_i = WT_{i-1} + (ST_i - ST_{i-1} - t_{i-1,i} - s_{i-1}), \tag{2}$$

$$FT_i = \min(FT_{i-1};\ l_i - ST_i + WT_i). \tag{3}$$

Besides, it is necessary to compute the latest start time LT_i of each node $i \in N_r$, to do so, we start calculating it from the last node $n \in N_r$, setting $LT_n = l_n$, and then proceed backward until the first node of the route as:

$$LT_{i-1} = \min(LT_i - t_{i-1,i} - s_i;\ l_{i-1}), \quad i = n, n-1, \ldots, 1. \tag{4}$$

Finally, the earliest and latest start times of the route r are calculated by:

$$est_r = e_1 + \min(FT_n;\ WT_n), \tag{5}$$

$$lst_r = LT_1. \tag{6}$$

5.2.1 Fixed Route Start Times

To solve the problem in which the starting time of the routes is fixed, we have developed the greedy algorithm described in 1. The input is a set of routes R resulting from the previous step, where the starting time of each route is already defined. Each route in R is assigned to a day $p \in P$, to a depot $d \in D$, and to a vehicle type $v \in V$. Furthermore, every route has a starting time α and a duration time β. The objective is to combine the routes of R in order to minimize the number of used vehicles per day.

In Algorithm 1, σ represents the set of used vehicles associated to the day p, the depot d and the vehicle type v (i.e., σ_{pdv}). We also have R_{pdv}, as the subset of routes in R that depart from depot d and are performed by a vehicle of type v, in day p.

Besides, the routes in R_{pdv} are sorted in non-decreasing order of starting time α. Following this order, we assign each route to an available vehicle in σ_{pdv}. In other words, a vehicle of type v assigned to a route $r \in R_{pdv}$ is a vehicle that can perform the route respecting its starting time and duration, as well as the vehicle operating hours T in day p and the time window of each customer.

Algorithm 1 Greedy Fixed Starting Times

```
procedure GreedyFixedST($R$)
    $\sigma \leftarrow \varnothing$
    for each day $p \in P$ do
        for each depot $d \in D$ do
            for each vehicle type $v \in V$ do
                Let $R_{pdv}$ be the subset of routes in $R$ associated to $p, d, v$
                Sort $R_{pdv}$ in non-decreasing order of $\alpha$
                for each $r \in R_{pdv}$ do
                    if the assignment is feasible then
                        $\sigma_{pdv} \leftarrow r$
                    end if
                end for
            end for
        end for
    end for
    return the number of used vehicles in $\sigma_p$ for each $p \in P$
end procedure
```

5.2.2 Bounded Route Start Times

5.2.3 Mathematical Model

The parameters above are used in the next mathematical model. The aim of the model is to combine the routes $r \in R$ to minimize the number of used vehicles per day. The model adopts three sets of binary decision variables and one set of continuous decision variables, which are defined in Table 1. Parameter T_r represents the total duration of route $r \in R$. This parameter takes into consideration the traveling time between the nodes in r, the fixed loading time, and the service time at each node in r. Parameter M represents a big number.

Table 1 Model decision variables

x_{rkv}	Binary variable taking the value 1 if route $r \in R$ is assigned to a vehicle $k \in K_v$ of type $v \in V$, 0 otherwise.
y_{kv}	Binary variable taking the value 1 if a vehicle $k \in K_v$ of type $v \in V$ is used, 0 otherwise.
z_{rskv}	Binary variable taking the value 1 if route $r \in R$ precedes route $s \in R$, and they are both assigned to the same vehicle $k \in K_v$ of type $v \in V$, 0 otherwise.
t_{rkv}	Continuous variable indicating the starting time of route $r \in R$ assigned to vehicle $k \in K_v$ of type $v \in V$.

The resulting mathematical model is given in (7)-(18) below. It is executed for each day $p \in P$, and so it only considers the routes $R_p \subseteq R$ that are performed on the day p.

$$Z_p = \quad \min \sum_{v \in V} \sum_{k \in K_v} y_{kv} \tag{7}$$

$$\text{s.t.} \quad x_{rkv} \leq y_{kv}, \quad \forall r \in R_p, \forall v \in V, \forall k \in K_v \tag{8}$$

$$\sum_{v \in V} \sum_{k \in K_v} x_{rkv} = 1, \qquad \forall r \in R_p \tag{9}$$

$$\sum_{r \in R_p} T_r x_{rkv} \leq T, \qquad \forall v \in V, \forall k \in K_v \tag{10}$$

$$z_{rskv} + z_{srkv} \geq x_{rkv} + x_{skv} - 1, \quad \forall v \in V, \forall k \in K_v, \forall r, s \in R_p : r \neq s \tag{11}$$

$$z_{rskv} + z_{srkv} \leq 1, \quad \forall v \in V, \forall k \in K_v, \forall r, s \in R_p : r \neq s \tag{12}$$

$$t_{rkv} + T_r \leq t_{skv} + M(1 - z_{rskv}), \quad \forall v \in V, \forall k \in K_v, \forall r, s \in R_p : r \neq s \tag{13}$$

$$est_r \leq t_{rkv} \leq lst_r, \quad \forall r \in R_p, \forall v \in V, \forall k \in K_v \tag{14}$$

$$x_{rkv} \in \{0, 1\}, \quad \forall r \in R_p, \forall v \in V, \forall k \in K_v \tag{15}$$

$$y_{kv} \in \{0, 1\}, \qquad \forall v \in V, \forall k \in K_v \tag{16}$$

$$z_{rskv} \in \{0, 1\}, \quad \forall r, s \in R_p : r \neq s, \forall v \in V, \forall k \in K_v \tag{17}$$

$$t_{rkv} \geq 0, \quad \forall r \in R_p, \forall v \in V, \forall k \in K_v \tag{18}$$

The objective function (7) aims to minimize the number of used vehicles. Constraints (8) ensure that each route r is assigned to a given vehicle k only if k performs that route. Constraints (9) guarantee that all routes are served by a vehicle. Constraints (10) ensure that multiple routes performed by a vehicle must be executed within the maximum vehicle working time T. Constraints (11) and (12) guarantee the precedence between routes that are performed by the same vehicle. In (11), if two routes are performed by the same vehicle, then one must precede the other. Instead, in (12), the first route precedes the second, or the second route precedes the first one. In constraints (13), if one route precedes another, then the second route must start only when the vehicle finishes servicing the first route, including the fixed loading time needed to satisfy the second route. Constraints (14) ensure that the starting time of each route is between the earliest and latest start time, computed using (5) and (6), respectively. Finally, constraints (15)-(18) define the variables domain.

5.2.4 Iterated Local Search

[31]

Algorithm 2 ILS algorithm

```
procedure ILS($I_{\max}$ = max number of iterations without improvements)
    ITERATION ← 0                    ▷ the number of iterations without improvement
    $s^* \leftarrow$ CONSTRUCTIVEHEURISTIC
    $s^* \leftarrow$ LOCALSEARCH($s^*$)
    while ITERATION $\leq I_{\max}$ do
        $s' \leftarrow$ PERTURBATION($s^*$)
        $s'' \leftarrow$ LOCALSEARCH($s'$)
        if ACCEPT($s^*, s''$) then
            $s^* \leftarrow s''$
            ITERATION ← 0
        else
            ITERATION ← ITERATION + 1
        end if
    end while
    return $s^*$
end procedure
```

6 Computational Results

** TO-DO: DA PARAFRASARE

We computationally evaluated the DSS on a real-world application encountered by Coopservice. The algorithms in Section 5.1 were coded in C++ and those in Section 5.2.3 in Python 3.8. Model (7)-(18) was solved by means of Coin-OR[4]. The computational experiments were executed on an Intel(R) Core(TM) i7-8750H CPU 2.20GHz, with 16 GB of RAM, running Microsoft Windows 11 Home 64-bits. A time limit of 10 seconds was imposed on each run.

The data of the real-world application have been obtained from the operations planned in the Sardinia region, Italy. The area is composed of 309 customers that are divided into two groups: North Sardinia, with 151 customers, and South Sardinia, with 158 customers. The North area is equipped with three depots, and the South with a single one. Each depot is equipped with two types of vehicles. The application has not started yet and is still in its initial planning phase. Minimizing the number of vehicles is thus important to establish the correct size and composition of the fleet.

For the creation of the scenario, geo-referencing and distance computation services have been used. With this data, the solver was then invoked to compute the optimized set of routes and the corresponding vehicles to be used. Figure 5 gives a graphical

[4] https://www.coin-or.org/documentation.html

representation of the area, where blue circles represent the customers and red hexagons the depots. The figure has been obtained using QGIS[5].

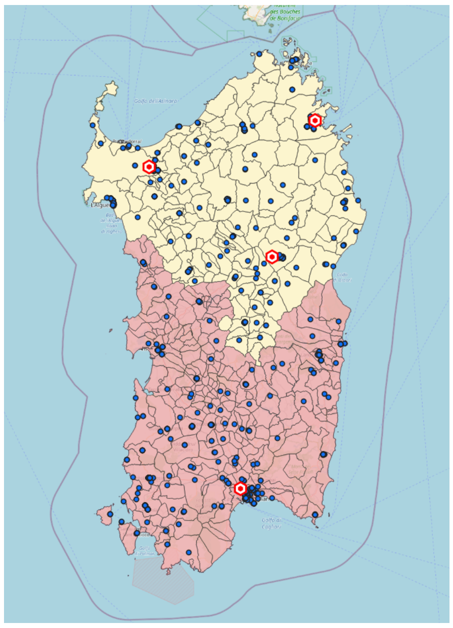
Fig. 5 Coopservice data of the Sardinia region

The aim is to generate a weekly schedule of the routes and minimize the number of used vehicles through the model proposed in this paper.

We consider 6 working days for the North and for the South, thus obtaining a total of 12 real instances. Table 2 reports the results that we obtained, for each day p and each area. Column $|R_p|$ indicates the number of routes generated by solving the VRPTW. Column Z_p reports the number of used vehicles per day p obtained after solving the MTVRPTW. Column $t(s)$ reports the computing times in seconds. In the first group of instances (North Sardinia), the algorithm reduces the number of vehicles from 49 to 46, and in the second one (South Sardinia) from 55 to 50. The computing time is always below one second per instance. It is important to take into consideration that, in Italy, the cost of a large vehicle (more than 120 tons) can easily exceed 50.000 euros. Therefore, even if the routes reduction (3 in the North and 5 in the South) may appear small at a first glance, they indeed represent a significant cost saving for the company.

To obtain a more extensive validation of the algorithm, we have created additional random instances based on the real ones. To this aim, we generated 20 weekly instances in the following way:

[5]https://www.qgis.org/it/docs/index.html

Table 2 Computational results on the real instances

Instance	p	$\|R_p\|$	Z_p	t(s)
North Sardinia	1	8	8	0.322
North Sardinia	2	9	9	0.437
North Sardinia	3	9	9	0.425
North Sardinia	4	10	9	0.605
North Sardinia	5	10	8	0.612
North Sardinia	6	3	3	0.038
Total		49	46	2.440

Instance	p	$\|R_p\|$	Z_p	t(s)
South Sardinia	1	10	9	0.358
South Sardinia	2	11	10	0.430
South Sardinia	3	11	9	0.933
South Sardinia	4	9	8	0.295
South Sardinia	5	9	9	0.270
South Sardinia	6	5	5	0.049
Total		55	50	2.334

- Group 1: it consists again of North Sardinia, but this time the number of depots is randomly selected in the set $\{1, 2, 3\}$ and the number of customers is a multiple of 30, going from 30 to 150. All customers are randomly selected from the original 151 customers in the real instance. We assume that there are two types of vehicles available in each depot, as in the original instance. The customers are randomly divided into six working days and their demand is coincident with the one they had in the original instance;
- Group 2: it is equivalent to Group 1 but refers to South Sardinia. In this case, all instances have one depot, two types of vehicles, six working days, and 30, 60, 90, 120, or 150 customers divided in the working days.

Table 3 presents the aggregate computational results obtained on the 20 random weekly instances. Each line gives total values over the six runs that have been executed (one per working day). The first three columns report the name of the instances, the number $|D|$ of depots, and the overall number $|C|$ of customers in the week. The total number of VRPTW routes is indicated by $|R|$ and is computed as $|R| = \sum_p |R_p|$. Similarly, the total number of MTVRPTW vehicles is indicated by Z and is computed as $Z = \sum_p Z_p$. The difference between the two values is reported in column Δ, with $\Delta = R - Z$. The rightmost column, $t(s)$, gives the total computing time in seconds over the six runs.

By looking at the results, we observe that no improvement has been obtained in three instances (namely, Inst-02, Inst-05, and Inst-06), all of which refer to North Sardinia. For all other instances, instead, the optimization algorithm managed to decrease the number of used vehicles. The improvement is equal to 2.15 on average and raises to 5 for the last instance. Notably, the computing time is always below three seconds.

Table 3 Computational results on the random instances

Instance	$\|D\|$	$\|C\|$	$\|R\|$	Z	Δ	t(s)
Inst-01	1	30	18	16	2	0.226
Inst-02	1	60	23	23	0	0.263
Inst-03	1	90	38	35	3	0.641
Inst-04	1	120	45	42	3	1.083
Inst-05	1	150	49	49	0	1.190
Inst-06	2	30	13	13	0	0.158
Inst-07	2	60	28	27	1	0.460
Inst-08	2	90	31	29	2	0.550
Inst-09	2	120	42	38	4	1.181
Inst-10	2	150	49	46	3	1.739
Inst-11	3	30	15	14	1	0.235
Inst-12	3	60	29	27	2	0.742
Inst-13	3	90	32	30	2	0.813
Inst-14	3	120	39	36	3	1.334
Inst-15	3	150	49	46	3	2.440
Inst-16	1	30	18	16	2	0.197
Inst-17	1	60	29	25	4	0.382
Inst-18	1	90	39	38	1	0.688
Inst-19	1	120	43	41	2	0.840
Inst-20	1	150	55	50	5	2.334
Average			34.20	32.05	2.15	0.875

The DSS proposed in this paper integrates a visualization tool to plot the elaborated routes on a map. The visualization part is just a front-end for the underlying micro-services. The simplified user interface allows a non-expert user to easily visualize the solution and check the feasibility of the routes. As an example, Figure 6 reports the solution obtained for the real instance of North Sardinia, in which different colors indicate a different type of used vehicle.

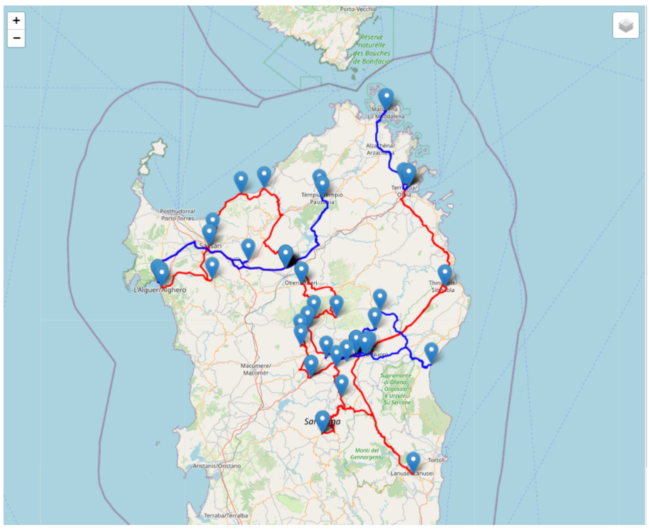
Fig. 6 Detail of a solution generated by the MTVRPTW model

****

6.1 Experimental setup

In order to evaluate the performance of the considered system we deployed the prototype DSS based on micro-service on a VM in private cloud environment. The VM for the DSS is equipped with 16GB of RAM and a 16 virtual cores. The underlying virtualization environment is based on XCP-NG (Xen-based) 8.2 and runs on a four Intel Xeon Gold 6252N CPU with a clock of 2.30 GHz.

For the experiments carried out on the DSS prototype, we consider several data sets with real routing problems provided by the industrial partner referring to several Italian regions such as Sardinia and Emilia-Romagna. Problem size is measured based on the number of geo-references points used in the problem (in the experiments we call these points Points of Interest or POI). In our experiments we perform scalability analyses. to this aim we derive sub-problem from the largest problem instances with the goal to explore how the performance are affected by the problem size.

6.2 Scalability analysis

In order to evaluate the impact of problem size on the problem, we consider the several steps carried out from geo-referencing the points to solving the routing problem, as described in Sec. 3.

The first experiment, shown in Fig. 7, shows the execution time of the geo-referencing of a set of addresses as a function to the set length. The sets are obtained as subsets of the initial set of data taken from the Emilia-Romagna data set. In these experiments, each resolution is repeated five times and average and standard deviation are computed. In order to be fair, the cache is cleaned before running each experiment.

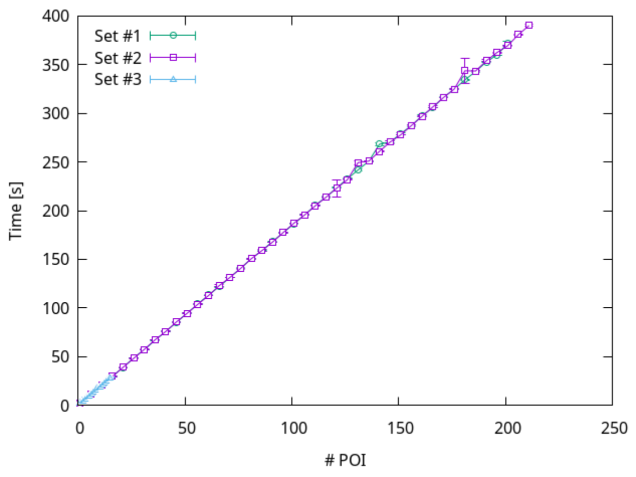

Fig. 7 Execution time of geo-location service

In Fig. 7, we observe an almost linear increase of the the time for resolution as the size of the data set grows. The performance on this area are constrained by the execution time of the external ArcGIS service used for geo-referencing. Due to the nature of the service, where a single key is used to access the external service, multi-threaded and parallel execution of the service do not provide a significant gain (the invocation rate for a given key is limited).

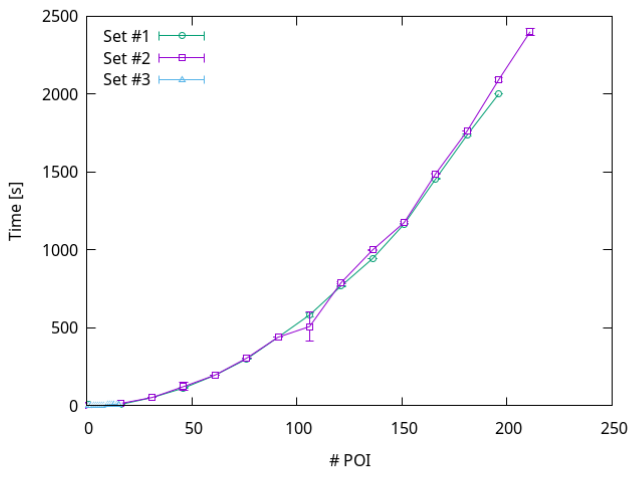

Fig. 8 Execution time of distance-computation service

The second set of the experiments experiments concerns the performance and scalability of the distance matrix computation. To this aim, we consider the output of the previous step and we feed the geo-referenced point of interests into the distance computation micro-service. For each problem size we record the time to obtain the distance matrix.

Fig. 8 shows the micro-service execution time as a function of the problem size. We recall that the distance matrix contains a value for every couple of points. This explains the quadratic shape f the execution time curve as a function of the number of problems. We also observe that, due to the constrain in the frequency of external service invocations, the distance computation matrix can require a time in the order of tens of minutes. This observation confirms the validity of our design choice to return a job ID for subsequent polling rather than returning directly the distance at the end of the computation: the timeout for the TCP connection supporting the micro. service interaction would return a timeout error before completing the service whenever the execution time of the micro-service exceeds 120 seconds. A second critical message from these experiments is the critical impact of caching on performance: whenever a distance is already available in the cache, there is no need to contact the external service.

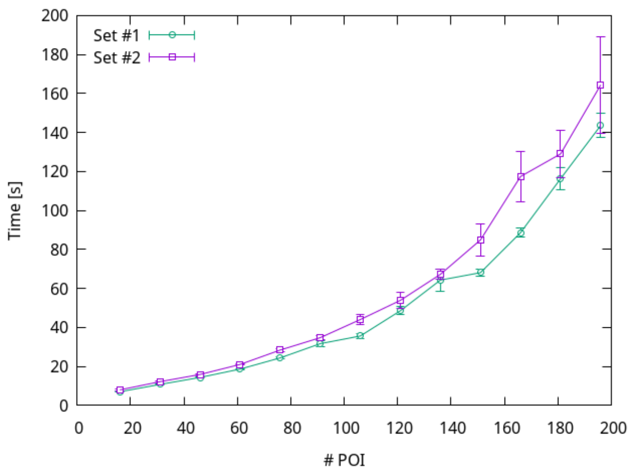

Fig. 9 Execution time of routing service

A third experiment concern the execution time of the vehicle routing problem that is the first step of the solver. Again we evaluate the scalability of the micro-service as a function of the problem size. Fig. 9 shows the performance of the Kramer algorithm implementation. We observe a non-linear increase of the execution time as a function of the number of points to visit, that is compatible with the $\mathcal{NP}$ nature of the vehicle routing problem. However, the limited size of the considered problem instances makes it feasible to provide an approximate solution of the problem in a reasonable time.

*** Qui servono i dati degli altri esperimenti ***

7 Concluding Remarks

** TO-DO: DA PARAFRASARE

Decision support systems (DSS) are becoming more popular in companies. This paper described the development of a model-driven DSS aimed at helping decision-makers in dealing with complex transportation problems. The proposed DSS contains modules to solve vehicle routing problems, particularly the multi-trip vehicle routing problem with time windows (MTVRPTW). Due to the complexity of the tasks involved, the operations are split into several micro-services that can geo-reference data and compute distances (interacting efficiently with external services that have a strict bound on the invocation rate). The obtained information is then used to solve the MTVRPTW, which is decomposed into two steps. In the first step, we use a metaheuristic algorithm to solve the VRPTW, and in the second step, we propose a mathematical model that redefines the route start times and solves the MTVRPTW. It is worth noting that the overall DSS architecture allows a modular evolution of its functions because data sources and even the solution heuristics can be easily changed.

The proposed approach has been tested on real and random instances with different numbers of depots and customers. Computational results highlight a reduction in the number of used vehicles with respect to the initial value obtained at the fist-step. This reduction brings significant benefits for the company in terms of logistics costs.

We observe that there is room for further improvements. First of all, the implemented model is able to find good solutions for up to 3 depots and 158 customers. The model could be replaced by a metaheuristic algorithm to solve larger instances. This represents the first interesting direction for future research. Further future research avenues in which we are interested are: adding new modules in the DSS, e.g., to handle new VRP variants, as for the car patrolling; improving the existing modules, e.g., proposing an integrated approach to solve the MTVRPTW, instead of a two-phase approach. The development of an integrated approach might also help us compare our application with the most sophisticated solution methods proposed in the literature (see, e.g., [9]). Again we point out that the micro-service architectural approach is a key feature to enable this evolution. Finally, we point out that we would like to apply this architecture to improve model-driven DSSs in other contexts, for example, the logistic activities related to the management of energy networks (see, e.g. [32]).

Acknowledgments. The authors would like to thank the support given by the National Council for Scientific and Technological Development (CNPq grants numbers 405369/2021-2 and 311185/2020-7), and the State of Goiás Research Foundation (FAPEG). We also thank Coopservice Soc.coop.p.A. for financial support and for sharing relevant information and knowledge for this research.

References

[1] Golden, B., Raghavan, S., Wasil, E. (eds.): The Vehicle Routing Problem: Latest Advances and New Challenges. Operations Research/Computer Science Interfaces, vol. 43. Springer, Boston, MA (2008). https://doi.org/10.1007/978-0-387-77778-8

[2] Toth, P., Vigo, D.: Vehicle Routing: Problems, Methods, and Applications. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (2014).

https://doi.org/10.1137/1.9781611973594

[3] Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., Subramanian, A.: New benchmark instances for the capacitated vehicle routing problem. European Journal of Operational Research **257**(3), 845–858 (2017) https://doi.org/10.1016/j.ejor.2016.08.012

[4] Kramer, R., Cordeau, J.-F., Iori, M.: Rich vehicle routing with auxiliary depots and anticipated deliveries: An application to pharmaceutical distribution. Transportation Research Part E: Logistics and Transportation Review **129**, 162–174 (2019) https://doi.org/10.1016/j.tre.2019.07.012

[5] Cavecchia, M., De Queiroz, T.A., Iori, M., Lancellotti, R., Zucchi, G.: A Decision Support System for Multi-Trip Vehicle Routing Problems. In: Proceedings of the 25th International Conference on Enterprise Information Systems, vol. 1, pp. 335–343. SCITEPRESS - Science and Technology Publications, Prague, Czech Republic (2023). https://doi.org/10.5220/0011806600003467

[6] Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. European Journal of Operational Research **231**(1), 1–21 (2013) https://doi.org/10.1016/j.ejor.2013.02.053

[7] Mendes, N., Iori, M.: A Decision Support System for a Multi-trip Vehicle Routing Problem with Trucks and Drivers Scheduling. In: Proceedings of the 22nd International Conference on Enterprise Information Systems, pp. 339–349. SCITEPRESS - Science and Technology Publications, Prague, Czech Republic (2020). https://doi.org/10.5220/0009364403390349

[8] Goel, R.K., Bansal, R.S.: Hybrid algorithms for rich vehicle routing problems: A survey. In: Nalepa, J. (ed.) Smart Delivery Systems. Intelligent Data-Centric Systems, pp. 157–184 (2020). Chap. 5. https://doi.org/10.1016/B978-0-12-815715-2.00011-7

[9] Vidal, T., Laporte, G., Matl, P.: A concise guide to existing and emerging vehicle routing problem variants. European Journal of Operational Research **286**(2), 401–416 (2020) https://doi.org/10.1016/j.ejor.2019.10.010

[10] Mor, A., Speranza, M.G.: Vehicle routing problems over time: a survey. Annals of Operations Research **314**(1), 255–275 (2022) https://doi.org/10.1007/s10479-021-04488-0

[11] Desrochers, M., Desrosiers, J., Solomon, M.: A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. Operations Research **40**(2), 342–354 (1992) https://doi.org/10.1287/opre.40.2.342

[12] Desaulniers, G., Madsen, O., Ropke, S.: The Vehicle Routing Problem with Time Windows. In: Toth, P., Vigo, D. (eds.) Vehicle Routing vol. 18, pp. 119–159 (2014).

Chap. 5. https://doi.org/10.1137/1.9781611973594.ch5

[13] Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: A unified solution framework for multi-attribute vehicle routing problems. European Journal of Operational Research **234**(3), 658–673 (2014) https://doi.org/10.1016/j.ejor.2013.09.045

[14] Amorim, P., Parragh, S.N., Sperandio, F., Almada-Lobo, B.: A rich vehicle routing problem dealing with perishable food: a case study. TOP **22**(2), 489–508 (2014) https://doi.org/10.1007/s11750-012-0266-4

[15] Keskin, M., Çatay, B.: A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. Computers & Operations Research **100**, 172–188 (2018) https://doi.org/10.1016/j.cor.2018.06.019

[16] Cattaruzza, D., Absi, N., Feillet, D.: Vehicle routing problems with multiple trips. Annals of Operations Research **271**(1), 127–159 (2018) https://doi.org/10.1007/s10479-018-2988-7

[17] Masmoudi, M.A., Hosny, M., Braekers, K., Dammak, A.: Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. Transportation Research Part E: Logistics and Transportation Review **96**, 60–80 (2016) https://doi.org/10.1016/j.tre.2016.10.002

[18] Wang, Z.: Delivering meals for multiple suppliers: Exclusive or sharing logistics service. Transportation Research Part E: Logistics and Transportation Review **118**, 496–512 (2018) https://doi.org/10.1016/j.tre.2018.09.001

[19] Pan, B., Zhang, Z., Lim, A.: Multi-trip time-dependent vehicle routing problem with time windows. European Journal of Operational Research **291**(1), 218–231 (2021) https://doi.org/10.1016/j.ejor.2020.09.022

[20] Nguyen, V.S., Pham, Q.D., Nguyen, T.H., Bui, Q.T.: Modeling and solving a multi-trip multi-distribution center vehicle routing problem with lower-bound capacity constraints. Computers & Industrial Engineering **172**, 108597 (2022) https://doi.org/10.1016/j.cie.2022.108597

[21] Power, D.J., Sharda, R.: Decision Support Systems. In: Nof, S.Y. (ed.) Springer Handbook of Automation, pp. 1539–1548. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-540-78831-7_87

[22] Farshidi, S., Jansen, S., Jong, R., Brinkkemper, S.: A decision support system for cloud service provider selection problem in software producing organizations. In: 2018 IEEE 20th Conference on Business Informatics (CBI), vol. 01, pp. 139–148 (2018). https://doi.org/10.1109/CBI.2018.00024

[23] Christoforou, A., Garriga, M., Andreou, A.S., Baresi, L.: Supporting the Decision of Migrating to Microservices Through Multi-layer Fuzzy Cognitive

Maps. In: Maximilien, M., Vallecillo, A., Wang, J., Oriol, M. (eds.) Service-Oriented Computing, pp. 471–480. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69035-3_34

[24] Di Francesco, P., Malavolta, I., Lago, P.: Research on architecting microservices: Trends, focus, and potential for industrial adoption. In: 2017 IEEE International Conference on Software Architecture (ICSA), pp. 21–30 (2017). https://doi.org/10.1109/ICSA.2017.24

[25] Keenan, P.B., Jankowski, P.: Spatial Decision Support Systems: Three decades on. Decision Support Systems **116**, 64–76 (2019) https://doi.org/10.1016/j.dss.2018.10.010

[26] ESRI: ArcGIS API for Python. https://developers.arcgis.com/python/api-reference/arcgis.gis.toc.html Accessed 2023-08-08

[27] Taibi, D., Lenarduzzi, V., Pahl, C.: Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. IEEE Cloud Computing **4**(5), 22–32 (2017) https://doi.org/10.1109/MCC.2017.4250931

[28] Hunt, J.: Applying the Model-View-Controller Pattern. In: Hunt, J. (ed.) Guide to the Unified Process Featuring UML, Java and Design Patterns, pp. 235–252. Springer, London (2003). https://doi.org/10.1007/1-85233-856-3_14

[29] Santini, A., Schneider, M., Vidal, T., Vigo, D.: Decomposition strategies for vehicle routing heuristics. INFORMS Journal on Computing **35**(3), 543–559 (2023) https://doi.org/10.1287/ijoc.2023.1288

[30] Savelsbergh, M.W.P.: The Vehicle Routing Problem with Time Windows: Minimizing Route Duration. ORSA Journal on Computing **4**(2), 146–154 (1992) https://doi.org/10.1287/ijoc.4.2.146

[31] Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated Local Search: Framework and Applications. In: Gendreau, M., Potvin, J.-Y. (eds.) Handbook of Metaheuristics. International Series in Operations Research & Management Science, pp. 129–168. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-91086-4_5

[32] Bruck, B.P., Castegini, F., Cordeau, J.-F., Iori, M., Poncemi, T., Vezzali, D.: A Decision Support System for Attended Home Services. INFORMS Journal on Applied Analytics **50**(2), 137–152 (2020) https://doi.org/10.1287/inte.2020.1031