A Decision Support System for Multi-Trip Vehicle Routing Problems

Mirko Cavecchia[1][a], Thiago Alves de Queiroz[2][b], Manuel Iori[1][c], Riccardo Lancellotti[3][d] and Giorgio Zucchi[4,5][e]

[1]*Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, 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, 41125, Modena, Italy*

[4]*School of Doctorate E4E, University of Modena and Reggio Emilia, 41121, Modena, Italy*

[5]*R&D Department, Coopservice S.coop.p.a, 42122, Reggio Emilia, Italy*

Keywords: Decision Support System, Micro-Services, Multi-Trip Vehicle Routing Problem.

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.

1 INTRODUCTION

The vehicle routing problem (VRP) is a milestone problem in combinatorial optimization whose primary purpose is to find an optimal set of routes for a fleet of vehicles in order to visit a set of customers. The VRP can be used to optimize several real-world applications, including the distribution of goods to customers, internal and external logistics, and the transportation of people (Golden et al., 2008) (Toth and Vigo, 2014).

Countless works have been produced in the VRP literature, and a variety of generalizations have been proposed over the years to deal with the different constraints that can be encountered in real-world applications (Vidal et al., 2020).

One of the main problem variants is the VRP with time windows (VRPTW), which imposes the service of each customer to be executed within a given time interval, called a time window. To the best of our knowledge, the first exact method for the VRPTW was proposed by (Desrochers et al., 1992), who used a column generation approach. Since then, many different VRPTW applications have been addressed in the literature, for example, in the delivery of food (Amorim et al., 2014), in the recharging of electric vehicles (Keskin and Çatay, 2018), and in the delivery of pharmaceutical products (Kramer et al., 2019).

Another well-established variant is the multi-trip VRPTW (MTVRPTW), a problem in which each vehicle can perform multiple routes, each starting and ending at the depot, to better fit the customers' time windows. Very recently, (Mor and Speranza, 2022) surveyed the VRP, the VRPTW, the MTVRPTW, and many other variants, including periodic routing problems and inventory routing problems.

[a] https://orcid.org/0000-0002-4129-9876
[b] https://orcid.org/0000-0003-2674-3366
[c] https://orcid.org/0000-0003-2097-6572
[d] https://orcid.org/0000-0002-9470-8784
[e] https://orcid.org/0000-0002-5459-7290

Cavecchia, M., Alves de Queiroz, T., Iori, M., Lancellotti, R. and Zucchi, G.
A Decision Support System for Multi-Trip Vehicle Routing Problems.
DOI: 10.5220/0011806600003467
In *Proceedings of the 25th International Conference on Enterprise Information Systems (ICEIS 2023) - Volume 1*, pages 335-343
ISBN: 978-989-758-648-4; ISSN: 2184-4992

The problem solved in this work originates from a real application at Coopservice Soc.coop.p.A., an Italian service company operating in the delivery of products to customers in several fields. Optimization algorithms have already been created for this application. In detail, (Kramer et al., 2019) proposed a metaheuristic to solve a VRPTW with additional constraints, whereas (Mendes. and Iori., 2020) combined a VRPTW with the need of scheduling trucks and drivers and solved the resulting problem by means of a mathematical model. Both works proposed a tailored solution method for a problem but did not consider the issues derived from its application in practice.

In this article, we propose a decision support system (DSS) to help companies in solving routing problems by easily invoking optimization algorithms. The DSS contains an optimization module to solve the MTVRPTW. In addition, it also includes a module based on the open source routing machine (OSRM), to calculate distance and travel time matrices using OpenStreetMap data (Luxen and Vetter, 2011). It also embeds preprocessing modules, to be invoked before optimization starts, that consist of accurate data analysis to, e.g., find the latitude and longitude coordinates starting from addresses and map customers in the road network. The DSS is based on the use of micro-services to obtain high scalability, maintainability, and fault tolerance. It has already been used by Coopservice in a real-world VRP application, and it has helped the company find an effective low-cost weekly delivery strategy.

The remainder of the paper is organized as follows. In Section 2, the developed DSS is drawn in detail. Section 3 provides a formal description of the optimization problem addressed. In Section 4, our two-step approach for solving the problem is discussed. Section 5 reports the outcome of computational tests executed on real and randomly generated instances. Finally, Section 6 presents the concluding remarks.

2 DECISION SUPPORT SYSTEM

Decision support systems are computer-based information systems that facilitate decision-making and can be used in a variety of contexts, such as business, finance, healthcare, and education. DSSs can be categorized based on various criteria, including their scope and functionality. According to (Power and Sharda, 2009), some common categories of DSSs are communications-driven, data-driven, document-driven, knowledge-driven, and model-driven.

2.1 Business Process Overview

The proposed software framework is a model-driven DSS and is tailored to the specific business process we aim to support. For this reason, we now shortly describe the main tasks required to provide a practical solution to an MTVRPTW.

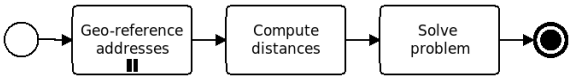

Figure 1: BPM description of the business process.

The basic structure of the business process is outlined in Figure 1 using a BPM notation. To take decisions, we start with a list of addresses that are the way-points of the trips. This list of addresses is geo-referenced (first box of the graph). The resulting list of coordinates is then used to create a distance matrix (travel times are used as the metric to express the distance between points). Finally, the distance matrix is used to define the MTVRPTW input that is solved using the proposed two-step approach.

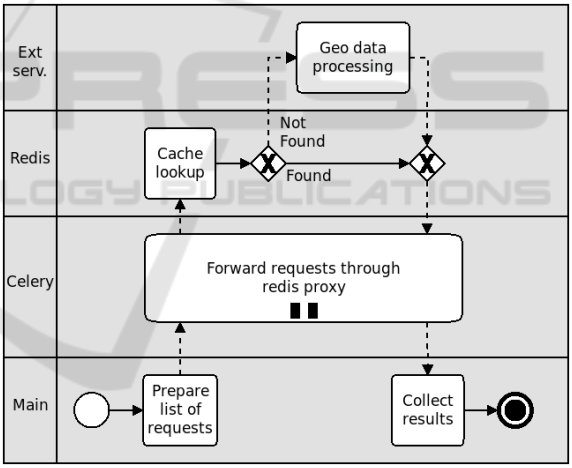

Figure 2: Address geo-referencing and distance computation.

Both geo-referencing and distance calculation are complex tasks that should be carried out relying on external services. However, such APIs are limited in the invocation rate. An interested reader can refer to https://developers.arcgis.com/python/api-reference/arcgis.gis.toc.html for more details. To this aim, in Figure 2 we outline how these tasks are carried out. First, the list of requests is prepared and submitted to a task queue implemented through Celery (https://docs.celeryq.dev/). A Celery task ID is used to identify a list of requests. This task ID is used to track the advancement of the ad-

dress resolution/distance calculation processes. The requests from Celery are not sent directly to the external APIs but are passed through a Redis object cache (https://redis.io/). Redis can store resolutions and distances in order to reduce the load on the external APIs and speed up the Celery tasks.

2.2 Services Definition

We now define the services that are combined to create the vehicle routing application. The services have been defined using an OpenAPI specification. However, for space reasons, only a short summary of the services is provided. Services are divided into three sets, according to the main components of the business process. The geo-referencing process includes the following tasks:

- Submission of the list of addresses to geo-reference. The supported input format includes also `.xls` files for compatibility with other tasks of the company. The submission of the address list returns an ID of the geo-referencing task;
- Task status. Given the task ID, the system returns the number of resolved addresses, in order to support user feedback on this task;
- Coordinates download. Once the task is completed, the list of coordinates can be retrieved. Both `JSON` and `.xls` output are supported. The former is to show points on a map (interacting with Open Street Maps APIs), the latter as an input for the next step to compute the distance matrix.

The structure of the APIs for the second task is similar, with the main difference that the main input is the `.xls` file with the coordinates and the output is an `.xls` file with the distance matrix. The task-based to monitor the progress is basically the same as the previously-described process.

The last step of the business process includes the two-step approach for the MTVRPTW. The main APIs can be summarized as:

- Problem resolution. The API invokes the solver algorithm that is implemented as a separate task that receives input from the Web APIs. The output of the algorithm is provided as the output of the API call. An additional handle is provided to guarantee access to the solver invocation at subsequent times;
- Access to solution data. This API uses the handle provided by the previous API to download the detailed solution. The solution can be exported both as a `JSON` structure, for visualization, and as an `.xls` file.

2.3 Technologies

To support companies in solving routing problems and to increase the knowledge and usability of optimization algorithms by non-expert users, we have developed an intuitive user-friendly web interface as a modular application based on micro-services.

A micro-services architecture is made up of small independent and loosely-coupled building blocks, each representing a service. This type of architecture has completely changed the way software is developed, making it extremely agile and leading to endless advantages over the monolithic architectures of the past, such as scalability, maintainability, and fault tolerance (Taibi et al., 2017).

The DSS is implemented through Docker, a widely used software platform to develop, test and deploy applications in a short time. Docker power is to package heterogeneous functionalities inside isolated images, called containers. These containers can be easily managed with high-level application programming interfaces (APIs) (https://docs.docker.com/).

The DSS is divided into two parts, the front-end, and the back-end. The first one deals with user interactions and is coded in React.JS, an open-source JavaScript framework used for web user interfaces. The second one is responsible for handling requests, computations, and data storage and is coded in Python using the Django web framework and model-view-controller paradigm (Hunt, 2003). The back-end part consists of four containers:

- App contains the optimization algorithms;
- OSRM is a C++ routing service designed to be run on OpenStreetMap data;
- Celery is an asynchronous task queue manager to share the task on different threads;
- Cache Redis is a service that manages the sending, receiving, and queuing of messages with Celery.

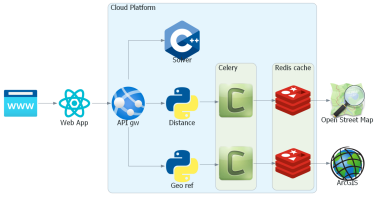

Figure 3: Web Interface Architecture.

The DSS has several operational features. Figure 3 shows a scheme of the DSS software architecture. The essential ones for our purposes are the following:

- Geo-reference: to geographically localize a list of addresses provided in input through an Excel file.

The respective output is an Excel file with the coordinates added to each address;

- Travel matrix generation: to create travel distance and time matrices starting from the coordinates of the previous module;
- MTVRPTW: to solve the MTVRP starting from customers' time slots, depots information, and vehicle types characteristics. Figure 4 shows a screenshot of the MTVRPTW module.

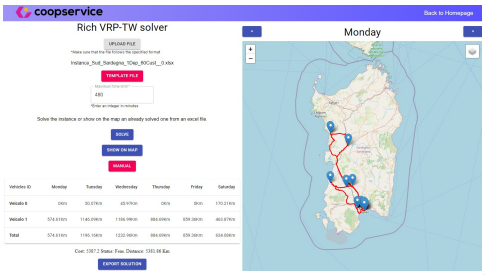

Figure 4: A screenshot of the MTVRPTW module.

3 PROBLEM DESCRIPTION

The MTVRPTW is formalized as follows. We are given a direct 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 is divided into depots (D) and customers (C), so that $N = D \cup C$. A traveling time t_{ij} is associated with each arc $(i, j) \in A$. A hard time window $[e_i, l_i]$ is associated with each node $i \in N$, where e_i is the earliest arrival time and l_i is the latest one. The vehicle visiting i cannot arrive after l_i, and it has to wait in case it arrives before e_i.

Each customer in C may require deliveries on multiple days. Let P denote the set of days in which we need to create the planning. Each customer $i \in C$ has a demand q_{ip} in day $p \in P$ and is associated with a service time s_i. The vehicle fleet is heterogeneous and divided into a set V of vehicle types. Each subset of vehicles of the same type is defined by K_v, and all vehicles $k \in K_v$ are identical to one another, that is, they have the same loading capacity and may travel along the same roads (e.g., mountainous arcs can be traveled only by the smallest vehicles).

A feasible solution for the problem must respect the following constraints: each route is associated with a unique depot and must respect the capacity of the vehicle; each vehicle can perform more than one route in a single day, but each route starts and ends at the depot; each customer is associated with exactly one route and their demand must be accomplished within a single visit; each customer must be visited inside their time window. In addition, the sum of the durations of the routes assigned to each vehicle should not exceed $T = 480$ minutes per day. Furthermore, we assume that, before starting any route, the vehicle has a fixed loading time of $\Delta = 30$ minutes (which must be included in the overall T limit).

The objective of the problem is to obtain a set of routes that satisfy the above constraints and minimize the number of used vehicles. All vehicles can operate on all days of set P. The problem is solved for all days $p \in P$, and the solution for a day is independent of the solutions for the other days.

4 SOLUTION APPROACH

To solve the MTVRPTW, we propose a two-phase method. In the first phase, we solve the VRPTW to obtain a set R of routes that satisfy the customers' demands, as explained in Section 4.1. In the second phase, we obtain a solution to the MTVRPTW, with a methodology that accepts variations in the start times of the routes, as described in Section 4.2. To this aim, we compute the earliest and latest possible start time of each route and then invoke a mathematical model to obtain the MTVRPTW solution.

4.1 Solving the VRPTW

(Kramer et al., 2019) solved a VRPTW based on a real-world distribution case study for Coopservice. The problem included a number of additional constraints, not described here in detail for the sake of conciseness (this type of problem is usually called rich VRP in the literature). The problem was related to the distribution of pharmaceutical products to hospitals and healthcare facilities, aiming to minimize the routing and warehouse costs. The authors proposed a metaheuristic algorithm and tested it on realistic and artificial instances. The realistic instances contain up to 232 nodes, and the artificial ones contain up to 300 nodes.

The algorithm by (Kramer et al., 2019) is a multi-start iterated local search, that starts from a solution obtained with a constructive heuristic, and improves it by means of local search and perturbation procedures. In the constructive heuristic, routes are created by adding customers to the closest depot with a greedy approach that inserts customers one at a time in the route that generates the lowest cost. Time windows can be violated but this induces a penalization in the objective function value.

The local search consists of a randomized variable neighborhood descent that has seven neighborhoods.

The neighborhoods are based on inter- and intra-route movements, like swap and relocation. The perturbation procedure is used to escape from local optima solutions by modifying the routes through random swap movements and customer relocations.

We use this heuristic to obtain a set R of routes to the MTVRPTW, but these routes do not yet take into consideration the fact that a vehicle can perform multiple trips in a single day. 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.

4.2 Solving the MTVRPTW

The proposed approach receives in input a set R of routes, each with its own starting time computed by the (Kramer et al., 2019) algorithm. 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, but obtaining more freedom in the possible assignment of multiple routes to the same vehicle. In this case, we face an optimization problem related to defining 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 $v \in V$, each depot $d \in D$, and each day $p \in P$. This is due to the fact that only routes that depart on the same day from the same depot and use the same vehicle type can be merged one with the other.

This optimization problem was solved by (Savelsbergh, 1992) through a forward-time slack procedure. The procedure is a key component of our approach, and we describe it in full detail next.

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:

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

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

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

We also need to compute the latest start time LT_i of each node $i \in N_r$. In this case, 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:

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

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

$$est_r = e_1 + \min(FT_n;\ WT_n), \quad (5)$$

$$lst_r = LT_1. \quad (6)$$

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.

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

s.t.

$$x_{rkv} \leq y_{kv}, \quad \forall r \in R_p, \forall v \in V, \forall k \in K_v \quad (8)$$

$$\sum_{v \in V} \sum_{k \in K_v} x_{rkv} = 1, \quad \forall r \in R_p \quad (9)$$

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

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

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

$$t_{rkv} + T_r \leq t_{skv} + M(1 - z_{rskv}),$$
$$\forall v \in V, \forall k \in K_v, \forall r,s \in R_p : r \neq s \quad (13)$$
$$est_r \leq t_{rkv} \leq lst_r,$$
$$\forall r \in R_p, \forall v \in V, \forall k \in K_v \quad (14)$$
$$x_{rkv} \in \{0,1\},$$
$$\forall r \in R_p, \forall v \in V, \forall k \in K_v \quad (15)$$
$$y_{kv} \in \{0,1\}, \qquad \forall v \in V, \forall k \in K_v \quad (16)$$
$$z_{rskv} \in \{0,1\},$$
$$\forall r,s \in R_p : r \neq s, \forall v \in V, \forall k \in K_v \quad (17)$$
$$t_{rkv} \geq 0,$$
$$\forall r \in R_p, \forall v \in V, \forall k \in K_v \quad (18)$$

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 COMPUTATIONAL RESULTS

We computationally evaluated the DSS on a real-world application encountered by Coopservice. The algorithms in Section 4.1 were coded in C++ and those in Section 4.2 in Python 3.8. Model (7)-(18) was solved by means of Coin-OR (https://www.coin-or.org/documentation.html). 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 representation of the area, where blue circles represent the customers and red hexagons the depots. The figure has been obtained using QGIS software (https://docs.qgis.org/3.22/en/docs/index.html).

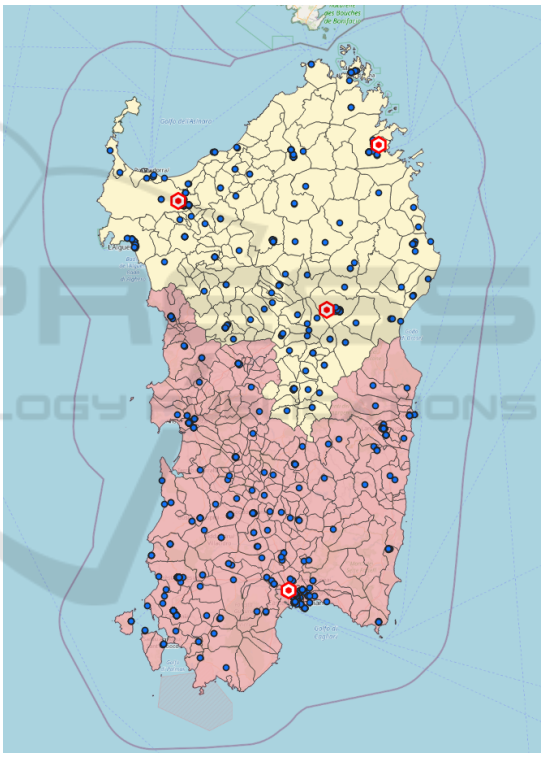

Figure 5: Coopservice data of the Sardinia region.

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.

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

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:

- 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.

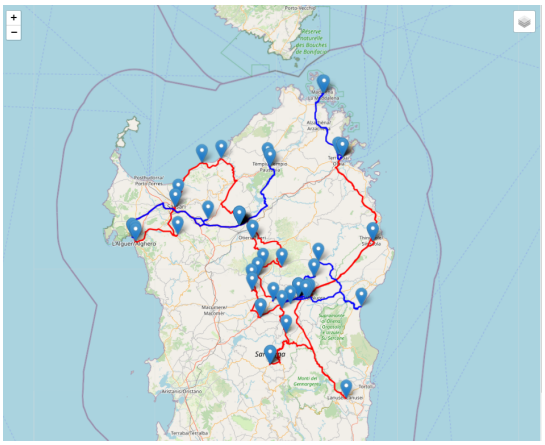

Figure 6: Detail of a solution generated by the MTVRPTW model.

6 CONCLUDING REMARKS

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., (Vidal et al., 2020)). 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. (Bruck et al., 2020)).

ACKNOWLEDGEMENTS

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

Amorim, P., Parragh, S. N., Sperandio, F., and Almada-Lobo, B. (2014). A rich vehicle routing problem dealing with perishable food: A case study. *TOP*, 22:489–508.

Bruck, B. P., Castegini, F., Cordeau, J.-F., Iori, M., Poncemi, T., and Vezzali, D. (2020). A decision support system for attended home services. *INFORMS Journal on Applied Analytics*, 50(2):137–152.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2):342–354.

Golden, B., Raghavan, S., and Wasil, E. (2008). *The Vehicle Routing Problem: Latest Advances and New Challenges*. Operations Research/Computer Science Interfaces Series. Springer.

Hunt, J. (2003). Applying the model-view-controller pattern. *Guide to the Unified Process featuring UML, Java and Design Patterns*, pages 235–252.

Keskin, M. and Çatay, B. (2018). A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. *Computers & Operational Research*, 100:172–188.

Kramer, R., Cordeau, J.-F., and Iori, M. (2019). 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.

Luxen, D. and Vetter, C. (2011). Real-time routing with openstreetmap data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, page 513–516, New York, NY, USA. Association for Computing Machinery.

Mendes., N. and Iori., M. (2020). 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 - Volume 1: ICEIS*, pages 339–349. SciTePress.

Mor, A. and Speranza, M. G. (2022). Vehicle routing problems over time: a survey. *Annals of Operations Research*, 314(1):255–275.

Power, D. J. and Sharda, R. (2009). Decision support systems. In *Shimon Y. Nof, eds., Springer Handbook of Automation*, pages 1539–1548. Springer Berlin Heidelberg.

Savelsbergh, M. W. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA journal on computing*, 4(2):146–154.

Taibi, D., Lenarduzzi, V., and Pahl, C. (2017). Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. *IEEE Cloud Computing*, 4(5):22–32.

Toth, P. and Vigo, D. (2014). *Vehicle routing: Problems, methods, and applications*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

Vidal, T., Laporte, G., and Matl, P. (2020). A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, 286(2):401–416.