

LEO TIŠLJARIĆ, M.Sc.<sup>1</sup>

E-mail: ltisljaric@fpz.unizg.hr

TONČI CARIĆ, Ph.D.<sup>1</sup>

E-mail: tcarić@fpz.unizg.hr

<sup>1</sup> Faculty of Transport and Traffic Sciences, University of Zagreb

Vukelićeva 4, 10000 Zagreb, Croatia, HR

## SOFTWARE TOOL FOR SOLVING THE VEHICLE ROUTING PROBLEM USING GOOGLE'S FRAMEWORK OR-TOOLS

### ABSTRACT

*This paper presents a software tool for solving the Vehicle Routing Problem (VRP). The software is designed as a web application with three main capabilities that include three modules: (i) data input module that includes choosing the depot and delivery locations on the map, (ii) module for solving the VRP problem, and (iii) visualization module for the assigned routes for every vehicle leaving the depot. For the implementation of the software, Django is used as a Python based framework for the backed, and Google's OR-Tools framework is used for the VRP solving.*

### KEY WORDS

*Vehicle Routing Problem (VRP); Python; Django; OR-Tools*

### 1. INTRODUCTION

The Vehicle Routing Problem (VRP) dates to the end of the fifties of the last century when Dantzig and Ramser set the mathematical programming formulation and algorithmic approach to solve the problem of delivering gasoline to service stations. Since then the interest in VRP evolved from a small group of mathematicians to the broad range of researchers and practitioners, from different disciplines, involved in this field today. The VRP definition states that  $K$  vehicles initially located at a depot are to deliver discrete quantities of goods to  $n$  customers (Figure 1). Determining the optimal route used by a group of vehicles when serving a group of users represents a VRP problem. The objective is to minimize the overall transportation cost. The solution of the classical VRP problem is a set of routes which all begin and end in the depot, and which satisfies the constraint that all the customers are served only once. The transportation cost can be improved by reducing the total travelled distance and by reducing the number of the required vehicles [1].

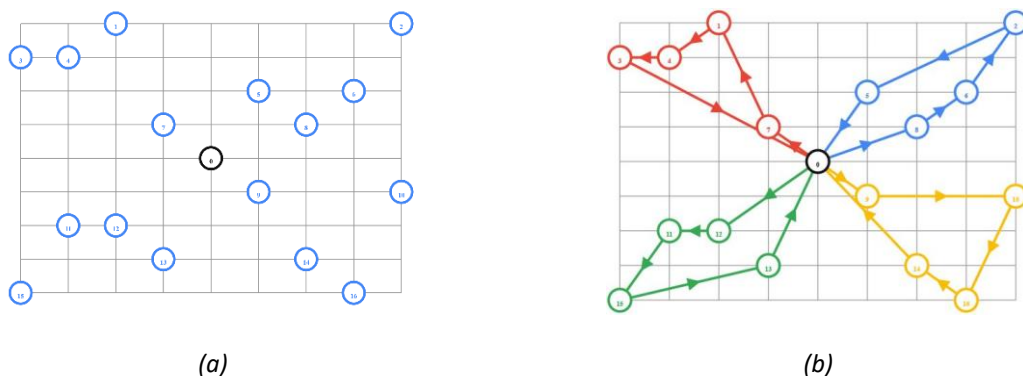


Figure 1 - Example of the VRP problem (a) with depot located on the index 0; solved VRP problem (b)

To be able to solve the VRP the traffic network must be represented as a directed graph  $G(E, V)$  where  $E$  represents the set of edges (road segments) and  $V$  represent the set of vertices (junctions).

Mathematically, the goal function of solving the VRP can be described as:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}, \quad (1)$$

where  $c_{ij}$  represents the cost of traveling from vertex  $i$  to vertex  $j$ , and the variable  $x_{ij}$  is a binary variable that shows if the edge bounded with vertices  $i$  and  $j$  will be considered for the optimal solution.

The constraints of a VRP can be summarized as follows:

$$\sum_{i \in V} x_{ij} = 1 \quad \forall j \in V \setminus \{0\}, \quad (2)$$

$$\sum_{j \in V} x_{ij} = 1 \quad \forall i \in V \setminus \{0\}, \quad (3)$$

$$\sum_{i \in V} x_{i0} = K, \quad (4)$$

$$\sum_{j \in V} x_{0j} = K, \quad (5)$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S), \quad \forall S \subseteq V \setminus \{0\}, \quad S \neq \emptyset, \quad (6)$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in V \quad (7)$$

where  $K$  represents the number of available vehicles and  $r(S)$  the minimum number of vehicles needed to serve set of customers  $S$ , with the assumption that depot is at vertex 0. Constraints (2) and (3) state that exactly one edge enters and exactly one leaves each vertex associated with a customer. Constraints (4) and (5) say that the number of vehicles leaving the depot is the same as the number entering it. Constraints (6) is the capacity cut constraints, which impose that the routes must be connected and that the demand on each route must not exceed the vehicle capacity. Finally, constraint (7) is the integrity constraint which explain that  $x_{ij}$  is a binary variable that have the value of 1 when an edge is considered for the optimal solution, 0 otherwise.

## 2. APPLICATION

Application is built using several frameworks. Django is used as a backend framework for the web application. Bootstrap is used as a framework for the frontend development, and the OR-Tools is a framework used for the VRP solving. This section briefly describes all frameworks and the structure of the application.

### 2.1 Used frameworks

- Django: a high-level Python Web framework. It enables the development and clean, pragmatic design. It is free and open source framework. It enables the building of the standard MVC (Model View Controller) web applications with connection to any available database [2].
- OR-Tools: an open source software suite for optimization, tuned solving problems in vehicle routing, flows, integer and linear programming, and constraint programming [3].

- Bootstrap: front-end open source toolkit enables quick design and customizable responsive mobile-first sites. It is popular because its simple grid-base approach and lot of pre-built features and classes [4].

## 2.2 Structure

The structure of the application and usage of the frameworks are described in Figure 2.

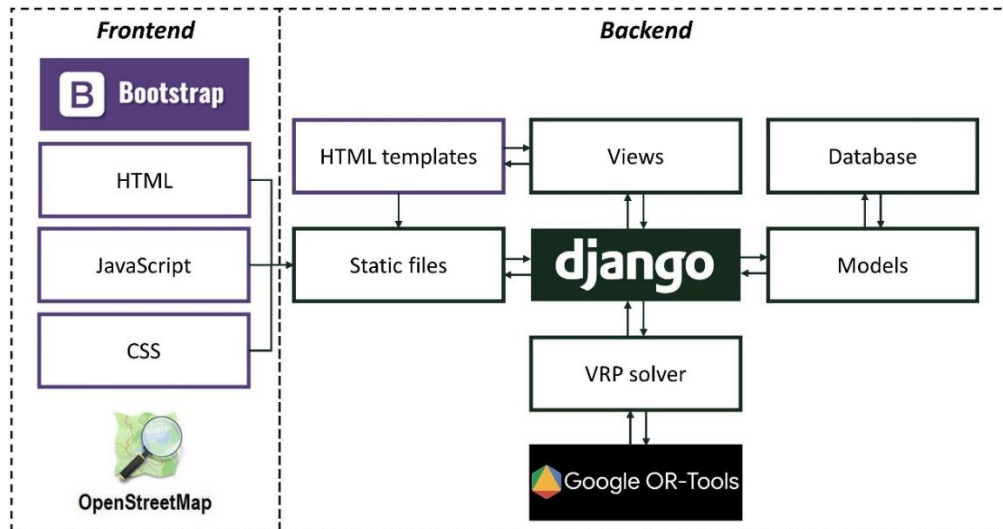


Figure 2 - Structure of the application with used frameworks

## 3. DESCRIPTION OF THE MODULES

The application consists of three modules: (i) data input module that includes choosing the depot and delivery locations on the map, (ii) module for solving the VRP problem, (iii) visualization module for the assigned routes for every vehicle leaving the depot, and (iv) administration module. In this section every module is briefly described.

### 3.1 Data input module

Data input module is used for entering the GPS coordinates of the depot and delivery places. Also, the object (model) *VrpProblem* is create and saved to database. The model contains basic information of the defined VRP and it defines the structure of the corresponding SQL table (Figure 3).

```

1 class VrpProblem(models.Model):
2     name = models.CharField(max_length=100)
3     depot_id = models.IntegerField()
4     description = models.CharField(max_length=250)
5     data_path = models.CharField(max_length=250, default="None")
6     def __str__(self):
7         return self.name
    
```

Figure 3 - VrpProblem module

The second model used in the data input module is *VrpPoint*. This model is used to save the GPS points in the database (Figure 4). The GPS points are retrieved using the Open Street Maps [5].

```
1 class VrpPoint(models.Model):
2     problem = models.ForeignKey(VrpProblem, on_delete=models.CASCADE)
3     lat = models.DecimalField(max_digits=30, decimal_places=26)
4     lon = models.DecimalField(max_digits=30, decimal_places=26)
5     point_id = models.IntegerField()
6     def __str__(self):
7         return "Problem id: " + str(self.problem) + " " + str(self.point_id)
```

Figure 4 - VrpPoint model

The frontend design of the data input module is shown in Figure 5. The design is simple and intuitive.

Figure 5 - Data input module – Main view

After the data input, click on the *Create VRP problem* button leads to output view presented in Figure 6. Output view shows if the operation of data input was successful and presents the basic parameters of the new input data.

Figure 6 - Data input module - Output view

### 3.2 Module for solving the VRP

This module's main purpose is solving the simple VRP problem using the Google's OR-Tools. The design of the view is shown in Figure 7. For solving the VRP, the problem name, number of vehicles, first solution strategy and the metaheuristic must be chosen.

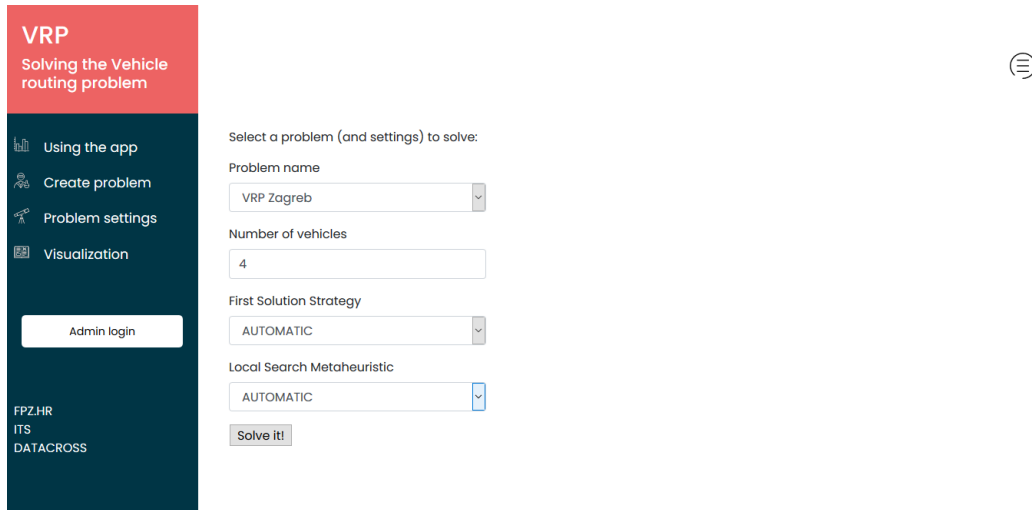
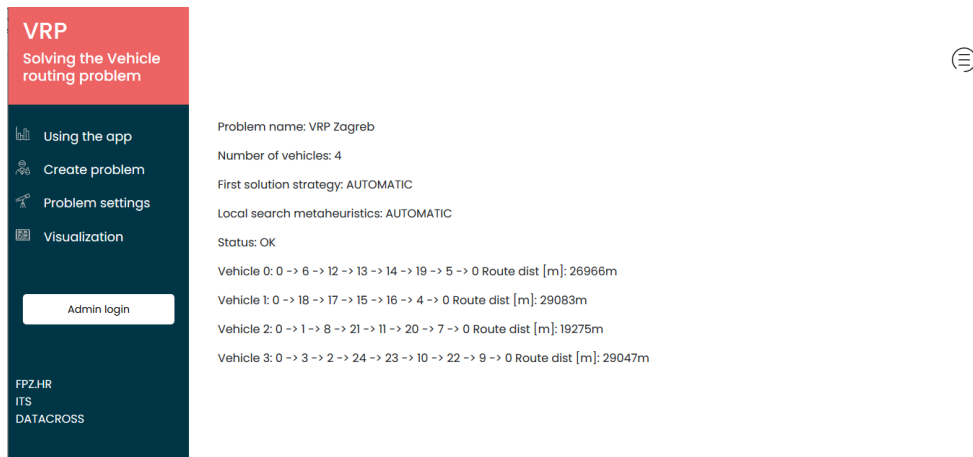


Figure 7 - Module for solving the VRP - Main view

After solving the VRP, output view is showed (Figure 8). This view shows the basic information about the solved problem and the routes for every vehicle.



Problem name: VRP Zagreb  
Number of vehicles: 4  
First solution strategy: AUTOMATIC  
Local search metaheuristics: AUTOMATIC  
Status: OK  
Vehicle 0: 0 -> 6 -> 12 -> 13 -> 14 -> 19 -> 5 -> 0 Route dist [m]: 26966m  
Vehicle 1: 0 -> 18 -> 17 -> 15 -> 16 -> 4 -> 0 Route dist [m]: 29083m  
Vehicle 2: 0 -> 1 -> 8 -> 21 -> 11 -> 20 -> 7 -> 0 Route dist [m]: 19275m  
Vehicle 3: 0 -> 3 -> 2 -> 24 -> 23 -> 10 -> 22 -> 9 -> 0 Route dist [m]: 29047m

Figure 8 - Module for solving the VRP - Output view

This module uses the model *RoutingPlan*. This model is used for saving the routing results in the SQL database (Figure 9).

```
1 class RoutingPlan(models.Model):
2     problem = models.ForeignKey(VrpProblem, on_delete=models.CASCADE)
3     vehicle_id = models.IntegerField()
4     routing_plan = models.TextField()
5     total_distance = models.IntegerField()
6     def __str__(self):
7         return "Problem: " + str(self.problem)
```

Figure 9 - RoutingPlan model

### 3.3 Visualization module

This module is used for the visualization of the VRP solution. On the main view (Figure 10) user must chose a problem from a database to visualize, and on the second view (Figure 11) the solution is shown on the map.

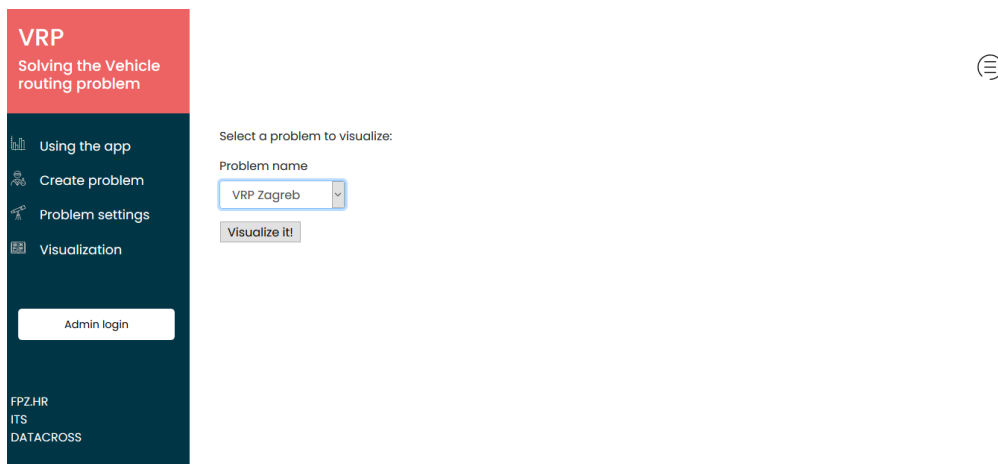


Figure 10 - Visualization module - Main view

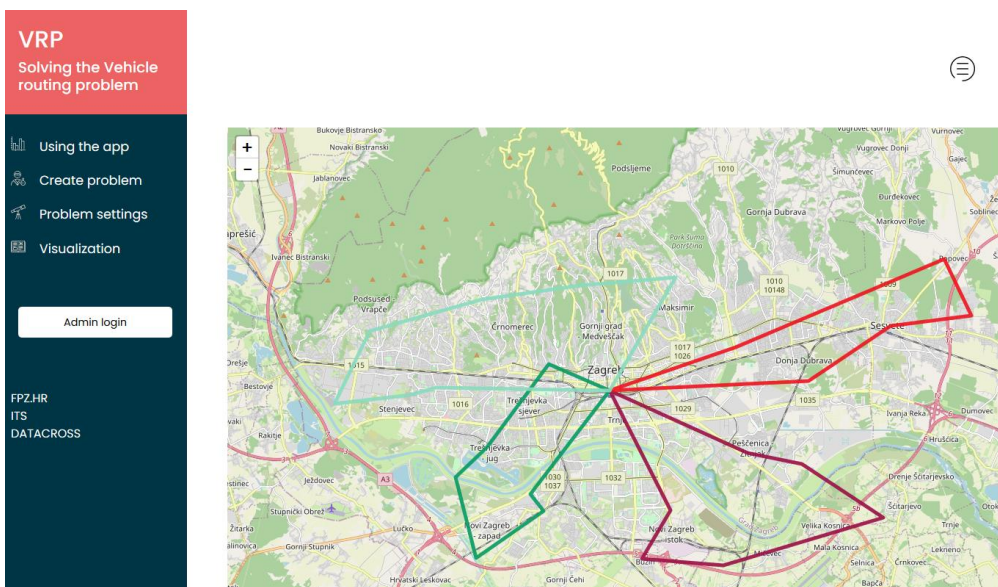


Figure 11 - Visualization module - Map view

### 3.4 Admin site

The administration site is shown in the Figure 12. The main purpose of this module is to manage the authentication on the site and to manage the database. The admin user can add, edit or delete users of the application and perform all necessary operations like CRUD on the database.



Figure 12 - Administration site - Main view

## 4. CONCLUSION AND FUTURE WORK

This paper presents a web-based application for solving the VRP. Application is build using the popular frameworks: i) Backend: Django, ii) Frontend: Bootstrap, JavaScript, and iii) VRP: OR-Tools.

Application have some limitations that need to bee addressed:

- Only simple VRP can be solved, without any constraints,
- Once solved, the VRP cannot be solved again with different settings,
- The VRP is solved using the Euclidean distances,
- Visualization is performed using the Euclidean distances.

Regarding the mentioned limitations, future work on the application is proposed:

- Using the road distances for solving the VRP and visualization,
- Alongside with distance, implement a traffic state of the link as a weight for the graph,
- Use another, more complex database,
- Add a choice of using some other solver like Gurobi, CPLEX, SCIP, GLPK or GLOP.

## REFERENCES

- [1] Carić T, Gold H. Vehicle Routing Problem. 1st ed. Vienna, Austria: I-Tech; 2008.
- [2] Django 1.5 [Internet]. Django Software Foundation, Lawrence, Kansas. 2013. Available from: <https://djangoproject.com>
- [3] Perron L, Furnon V. OR-Tools 7.2 [Internet]. Google. 2019. Available from: <https://developers.google.com/optimization/>

- [4] Bootstrap 4.1. [Internet]. 2020. Available from: <https://getbootstrap.com/>
- [5] Open Street Maps (OSM) [Internet]. OpenStreetMap contributors. 2017. Available from: <https://www.openstreetmap.org>