



Silesian  
University  
of Technology

**SILESIA N UNIVERSITY OF TECHNOLOGY**  
**FACULTY OF AUTOMATIC CONTROL, ELECTRONICS**  
**AND COMPUTER SCIENCE**

**PROGRAMME: INFORMATICS**

Final Project

Design and implementation of web application used for solving vehicle routing problems with time windows.

author: Łukasz Kwiecień

supervisor: Tomasz Jastrz ąb, PhD

Gliwice, December 2021



# Contents

<b>Abstract</b>	<b>1</b>
<b>1 Introduction</b>	<b>3</b>
1.1 Content outline . . . . .	4
<b>2 Problem analysis</b>	<b>7</b>
2.1 Vehicle Routing Problem . . . . .	7
2.2 Vehicle Routing Problem with Time Windows . . . . .	9
<b>3 Requirements and tools</b>	<b>13</b>
<b>4 External specification</b>	<b>15</b>
<b>5 Internal specification</b>	<b>17</b>
<b>6 Verification and validation</b>	<b>19</b>
<b>7 Conclusions</b>	<b>21</b>
<b>Bibliography</b>	<b>III</b>
<b>Index of abbreviations and symbols</b>	<b>VII</b>
<b>Listings</b>	<b>IX</b>
<b>List of additional files in electronic submission (if applicable)</b>	<b>XIII</b>
<b>List of figures</b>	<b>XV</b>



# Abstract

Abstract - the abstract text should be copied into the respective field in the APD system. Abstract with keywords should not exceed one page.

**Keywords: 2-5 keywords, separated by commas**



# Chapter 1

## Introduction

Transportation is one of the most critical activities in the supply chain. Its importance comes from the fact that the transportation costs can reach up to almost 40% of the total logistics costs of a manufacturing company. [6] For that reason, the companies need to transport the goods or persons efficiently. That goal can be achieved by either minimizing the distance traveled by the vehicles or reducing the number of routes required to visit all destinations. The Vehicle Routing Problem (VRP) describes the problem of assigning loads to the vehicles and sequencing the customers assigned to each vehicle to obtain optimal routes.

VRP is one of the most studied combinatorial optimization problems. Its popularity is due to the fact that VRP can be used in many real-life scenarios in the fields of distribution, collection, and logistics. In a VRP the fleet of vehicles has to visit a set of customers starting from a given depot and deliver commodities to them, taking into consideration all given constraints. There are various constraints that could be introduced to the problem, for example, the time windows for the customers or the capacity of vehicles.

Depending on the problem variant, VRP is a combination of two or more NP-hard problems, which makes VRP also NP-hard problem. The fact that the problem is NP-hard makes the process of finding an

exact solution very time-consuming. Therefore it is necessary to use a heuristic approach to get good solutions in an acceptable time.

In this thesis a web application solving Capacitated Vehicle Routing Problem with Time Windows (CVRTPW) is considered. User defines the problem by picking waypoints on the map and determining the capacity and time window constraints. Application, if feasible, solves the problem and presents the results in a user-friendly way by visualizing all routes on the map. The problem is solved using Push Forward Insertion Heuristic and optimized using Local Search with  $\lambda$  interchange method. The fleet of vehicles is homogeneous, which means that every vehicle has the same amount of goods that can be transported in it. Every customer has its own time window within which the delivery must be made. The goal was to create a tool that will simplify the process of defining and solving the CVRTPW for the end user.

## 1.1 Content outline

The second chapter "Problem analysis" provides history and explores scientific background material for the Capacitated Vehicle Routing Problem with Time Windows. Moreover provides formal and detailed definition of the problem and the solution methods based on the researched literature. Chapter 3 focuses on the design and implementation process of the project. Shows the functional and nonfunctional requirements as well as diagrams describing the system. The tools and methodologies used for the design and implementation process are also described there. The fourth chapter covers the hardware and software requirements for the application. Provides the step by step installation procedure along with the user manual. The usage examples and screenshots of the working application can also be found there. Chapter 5 discusses the software part of the project in detail. Description of the architecture, code and structure of the project is explained in this chapter. Furthermore specifies in detail the software-side of the entire



---

process of solving the CVRPTW. Chapter 6 shows how the project was tested and verified if the requirements set during the design process were fulfilled. The last chapter 'Conclusions' summarizes the achieved results and describes the encountered difficulties.



# Chapter 2

## Problem analysis

### 2.1 Vehicle Routing Problem

The problem concerns a set of customers to whom products need to be delivered in such a way that the delivery cost is as low as possible. Transport is performed by a fleet of vehicles, each of which can carry a certain number of goods. The goal is to serve all customers with as few vehicles as possible and to keep the total distance travelled by all vehicles as short as possible within predefined constraints. The classic version of VRP has the following constraints:

1. Each vehicle's route starts and ends at a depot.
2. All goods must be delivered.
3. Customer must receive all goods at one time delivered by one vehicle.
4. The vehicle may not carry more goods than its capacity allows.

The VRP can be formally defined as directed graph  $G = (V, A)$ , where  $V = \{0, \dots, n\}$  is the set of vertices representing customers and the depot, and  $A$  is the set of arcs  $(i, j)$  connecting these vertices. The set of vehicles is denoted by  $W$ . The depot is marked as vertex 0. The

number of vehicles is denoted by  $m$ , and each vehicle has a capacity of  $Q$ . The cost of travel between vertices  $i$  and  $j$  is denoted by  $c_{ij}$ . The cost is the distance, duration or other costs that may occur during the travel between nodes. The demand of customer  $i$  is represented as  $q_i$ . The customer subset  $S \subseteq V \setminus \{0\}$  can be served by a minimum of  $r(S)$  vehicles. This number can be obtained by solving the Bin Packing Problem (BPP) with bin set  $S$  with capacities  $Q$ , but because BPP is an NP-hard problem, it can be approximated by its lower bound:  $\lceil \sum_{i \in S} q_i / Q \rceil$ . [2]

The Integer Linear Programming formulation of the problem:[7]

$$\text{Minimize } \sum_{i,j \in V} c_{ij} x_{ij} \quad (2.1)$$

*subject to:*

$$\sum_{i \in V} x_{ij} = 1 \quad \forall j \in V \setminus \{0\} \quad (2.2)$$

$$\sum_{j \in V} x_{ij} = 1 \quad \forall i \in V \setminus \{0\} \quad (2.3)$$

$$\sum_{j \in V} x_{0j} = m \quad (2.4)$$

$$\sum_{i \in V} x_{i0} = m \quad (2.5)$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq \lceil \sum_{i \in S} q_i / Q \rceil \quad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset \quad (2.6)$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in V \quad (2.7)$$

In this formula,  $x_{ij}$  is a binary variable whose value indicates whether the arc between vertices  $i$  and  $j$  is traversed in the solution. If the connection between these vertices belongs to the solution, then the variable

$x_{ij}$  is equal to 1, otherwise it is equal to 0. *Indegree* (2.2) and *outdegree* (2.3) constraints guarantee that every customer is visited only once. Constraints 2.4 and 2.5 ensure that the solution has  $m$  routes, one for each vehicle (the number of routes does not exceed the number of vehicles). Constraints 2.6 are called *capacity-cut constraints* and refer to the minimum number of vehicles needed to serve a set of customers whose total demand is  $\sum_{i \in S} q_i$ . These constraints ensure the connectivity of the solution and that there are no paths whose load would exceed the capacity of the vehicle. The last constraint 2.7 ensures the binarity of the variable  $x_{ij}$ . If  $x_{0j}$  is equal to 1 it means that  $j$  belongs to this route. Any route from the solution can be reconstructed in this way: the next customer in the route is customer  $i$ , for whom  $x_{ij}$  is equal to 1. The last customer is customer  $i$  for whom  $x_{i0}$  is equal to 1. [7]

The problem of vehicle routing was first introduced by Dantzig and Ramser in 1959 as "The truck dispatching problem". [3] In the following years it became very popular, resulting in many studies, books and scientific publications exploring the topic. The problem is one of the most studied combinatorial optimisation problems, since it offers many benefits for logistics and transport companies. Different implementations can save as much as 5% to 30% of transport-related costs. [4]

## 2.2 Vehicle Routing Problem with Time Windows

The Vehicle Routing Problem with Time Windows (VRPTW) is a variant of VRP in which a constraint due to time windows is added. In this variant of the problem, each customer  $i$  has a time window  $[a_i, b_i]$  in which it must be served. A customer cannot be served earlier than  $a_i$  and later than  $b_i$ . In case the vehicle arrives earlier than  $a_i$  it must wait until the window opens. Moreover, to each arc  $(i, j)$  the time  $t_{ij}$  is assigned. For each vertex  $i$  and vehicle  $k$  a decision variable  $s_{ik}$  is defined.

This variable represents the time at which the vehicle  $k$  starts to serve the customer  $i$ . [1] This variant of VRP is also an NP-hard problem. [5]

The VRPTW can be stated mathematically as multicommodity network flow formulation with time windows and capacity constraints:[1]

$$\min \sum_{k \in W} \sum_{i, j \in V} c_{ij} x_{ijk} \text{ s.t.} \quad (2.8)$$

*subject to:*

$$\sum_{k \in W} \sum_{i \in V} x_{ijk} = 1 \quad \forall i \in V \setminus \{0\} \quad (2.9)$$

$$\sum_{i \in V \setminus \{0\}} q_i \sum_{j \in V} x_{ijk} \leq Q \quad \forall k \in W \quad (2.10)$$

$$\sum_{j \in V} x_{0jk} = 1 \quad \forall k \in W \quad (2.11)$$

$$\sum_{i \in V} x_{ihk} - \sum_{j \in V} x_{hjk} = 0 \quad \forall h \in V \setminus \{0\}, \forall k \in W \quad (2.12)$$

$$\sum_{i \in V} x_{i0k} = 1 \quad \forall k \in W \quad (2.13)$$

$$x_{ijk}(s_{ik} + t_{ij} - s_{jk}) \leq 0 \quad \forall i, j \in V, \forall k \in W \quad (2.14)$$

$$a_i \leq s_{ik} \leq b_i \quad \forall i \in V, \forall k \in W \quad (2.15)$$

$$x_{ijk} \in 0, 1 \quad \forall i, j \in V, \forall k \in W \quad (2.16)$$

Objective function (2.8) minimizes travel cost. Constraints (2.9) and (2.10) ensures that each customer is visited once and vehicle cannot surpass its capacity. Equations (2.11), (2.12) and (2.13) indicate that each vehicle must leave the depot, once it has reached the customer

---

it must travel to the next location and finally return to the depot. Inequality (2.14) shows the relationship between vehicle departure time and its successor. Inequalities (2.15) provide that the time windows are observed. Constraints (2.16) are the integrality constraints. [1]





# Chapter 3

## Requirements and tools

- functional and nonfunctional requirements
- use cases (UML diagrams)
- description of tools
- methodology of design and implementation



# Chapter 4

## External specification

- hardware and software requirements
- installation procedure
- activation procedure
- types of users
- user manual
- system administration
- security issues
- example of usage
- working scenarios (with screenshots or output files)



Figure 4.1: Figure caption (below the figure).



# Chapter 5

## Internal specification

- concept of the system
- system architecture
- description of data structures (and data bases)
- components, modules, libraries, resume of important classes (if used)
- resume of important algorithms (if used)
- details of implementation of selected parts
- applied design patterns
- UML diagrams

Use special environment for inline code, eg **descriptor** or **descriptor\_gaussian**  
. Longer parts of code put in the figure environment, eg. code in Fig.  
5.1. Very long listings—move to an appendix.

---

```
1 class descriptor_gaussian : virtual public descriptor
2 {
3     protected:
4         /** core of the gaussian fuzzy set */
5         double _mean;
6         /** fuzzyfication of the gaussian fuzzy set */
7         double _stddev;
8
9     public:
10        /** @param mean core of the set
11                @param stddev standard deviation */
12        descriptor_gaussian (double mean, double stddev);
13        descriptor_gaussian (const descriptor_gaussian & w
14            );
15        virtual ~descriptor_gaussian();
16        virtual descriptor * clone () const;
17
18        /** The method elaborates membership to the
19                gaussian fuzzy set. */
20        virtual double getMembership (double x) const;
21    };
```

---

Figure 5.1: The `descriptor_gaussian` class.

# Chapter 6

## Verification and validation

- testing paradigm (eg V model)
- test cases, testing scope (full / partial)
- detected and fixed bugs
- results of experiments (optional)





# Chapter 7

## Conclusions

- achieved results with regard to objectives of the thesis and requirements
- path of further development (eg functional extension ...)
- encountered difficulties and problems

Table 7.1: A caption of a table is **above** it.

$\zeta$	method						
	alg. 1	alg. 2	alg. 3			alg. 4, $\gamma = 2$	
			$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\beta = 0.1$	$\beta = -0.1$
0	8.3250	1.45305	7.5791	14.8517	20.0028	1.16396	1.1365
5	0.6111	2.27126	6.9952	13.8560	18.6064	1.18659	1.1630
10	11.6126	2.69218	6.2520	12.5202	16.8278	1.23180	1.2045
15	0.5665	2.95046	5.7753	11.4588	15.4837	1.25131	1.2614
20	15.8728	3.07225	5.3071	10.3935	13.8738	1.25307	1.2217
25	0.9791	3.19034	5.4575	9.9533	13.0721	1.27104	1.2640
30	2.0228	3.27474	5.7461	9.7164	12.2637	1.33404	1.3209
35	13.4210	3.36086	6.6735	10.0442	12.0270	1.35385	1.3059
40	13.2226	3.36420	7.7248	10.4495	12.0379	1.34919	1.2768
45	12.8445	3.47436	8.5539	10.8552	12.2773	1.42303	1.4362
50	12.9245	3.58228	9.2702	11.2183	12.3990	1.40922	1.3724

# Bibliography

- [1] Kallehauge B. et al. “Vehicle Routing Problem with Time Windows.” In: *Column Generation*. Boston, MA.: Springer, 2005, pp. 67–98.
- [2] J. F. Cordeau et al. “Vehicle Routing”. In: *Management Science* 14 (2007), pp. 367–428.
- [3] G. B. Dantzig and J. H. Ramser. “The Truck Dispatching Problem”. In: *Management Science* 6.1 (1959), pp. 80–91.
- [4] Geir Hasle, Knut-Andreas Lie and Ewald Quak. *Geometric Modelling, Numerical Simulation, and Optimization: Applied Mathematics at SINTEF*. Berlin: Springer, 2007. ISBN: 978-3-540-68783-2.
- [5] M. W. P. Savelsbergh. “Local search in routing problems with time windows”. In: *Annals of Operations Research* 4 (1985), pp. 285–305.
- [6] Katarzyna Sukiennik. “Logistics Costs in Manufacturing Companies”. In: *Zeszyty Naukowe Politechniki Częstochowskiej. Zarządzanie* 4 (2011), pp. 131–139.
- [7] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. Society for Industrial & Applied Mathematics, 2002. ISBN: 978-0-89871-498-2.



# Appendices



# Index of abbreviations and symbols

DNA deoxyribonucleic acid

MVC model–view–controller

$N$  cardinality of data set

$\mu$  membership function of a fuzzy set

$\mathbb{E}$  set of edges of a graph

$\mathcal{L}$  Laplace transformation





# Listings

(Put long listings in the appendix.)

---

```
1 partition fcm_possibilistic::doPartition
2                                     (const dataset & ds)
3 {
4     try
5     {
6         if (_nClusters < 1)
7             throw std::string ("unknown_number_of_clusters
8                                 ");
9         if (_nIterations < 1 and _epsilon < 0)
10            throw std::string ("You_should_set_a_maximal_
11                                number_of_iteration_or_minimal_difference_
12                                --epsilon.");
13         if (_nIterations > 0 and _epsilon > 0)
14            throw std::string ("Both_number_of_iterations_
15                                and_minimal_epsilon_set--you_should_set_
16                                either_number_of_iterations_or_minimal_
17                                epsilon.");
18
19         auto mX = ds.getMatrix();
20         std::size_t nAttr = ds.getNumberOfAttributes();
21         std::size_t nX     = ds.getNumberOfData();
22         std::vector<std::vector<double>> mV;
23         mU = std::vector<std::vector<double>> (_nClusters
```

```
    );  
18     for (auto & u : mU)  
19         u = std::vector<double> (nX);  
20     randomise(mU);  
21     normaliseByColumns(mU);  
22     calculateEtas(_nClusters, nX, ds);  
23     if (_nIterations > 0)  
24     {  
25         for (int iter = 0; iter < _nIterations; iter  
                ++)  
26         {  
27             mV = calculateClusterCentres(mU, mX);  
28             mU = modifyPartitionMatrix (mV, mX);  
29         }  
30     }  
31     else if (_epsilon > 0)  
32     {  
33         double frob;  
34         do  
35         {  
36             mV = calculateClusterCentres(mU, mX);  
37             auto mUnew = modifyPartitionMatrix (mV, mX)  
38                 ;  
39             frob = Frobenius_norm_of_difference (mU,  
                mUnew);  
40             mU = mUnew;  
41         } while (frob > _epsilon);  
42     }  
43     mV = calculateClusterCentres(mU, mX);  
44     std::vector<std::vector<double>> mS =  
        calculateClusterFuzzification(mU, mV, mX);  
45
```

74



# List of additional files in electronic submission

Additional files uploaded to the system include:

- source code of the application,
- test data,
- a video showing how software or hardware developed for thesis is used
- etc.



# List of Figures

4.1	Figure caption (below the figure). . . . .	15
5.1	The <b>descriptor_gaussian</b> class. . . . .	18





# List of Tables

7.1	A caption of a table is above it. . . . .	22
-----	---	----