Silesian
University
of Technology

SILESIAN 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 Jastrząb, PhD

Gliwice, November 2021

Contents

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. [1] 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 determines the capacity and time window constraints. Application, if feasible, solves the problem and presents the results in the 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 λ interchange method. The fleet of vehicles is homogeneus, 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 nonfuctional requirements as well as diagrams describing the system. The tools and metodologies 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]

- problem analysis
- state of the art, problem statement
- literature research (all sources in the thesis have to be referenced [bib:article, bib:book, 2, 3])
- description of existing solutions (also scientific ones, if the problem is scientifically researched), algorithms, location of the thesis in the scientific domain

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.

```
class descriptor_gaussian : virtual public descriptor
{
   protected:
      /** core of the gaussian fuzzy set */
      double _mean;
      /** fuzzyfication of the gaussian fuzzy set */
      double _stddev;

   public:
      /** @param mean core of the set
          @param stddev standard deviation */
      descriptor_gaussian (double mean, double stddev);
      descriptor_gaussian (const descriptor_gaussian & w
         );
      virtual ~descriptor_gaussian();
      virtual descriptor * clone () const;

      /** The method elaborates membership to the
         gaussian fuzzy set. */
      virtual double getMembership (double x) const;

};
```

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.

	method						
			alg. 3			alg. 4, $\gamma = 2$	
ζ	alg. 1	alg. 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] Katarzyna Sukiennik. “Logistics Costs in Manufacturing Companies”. In: *Zeszyty Naukowe Politechniki Częstochowskiej. Zarządzanie* —.4 (2011), pp. 131–139.

[2] Name Surname, Name Surname and N. Surname. “Title of a conference article”. In: *Conference title*. 2006, pp. 5346–5349.

[3] Name Surname, Name Surname and N. Surname. *Title of a web page*. `http://somewhere/in/internet.html`. [access date: 2018-09-30].

Appendices

Index of abbreviations and symbols

DNA	**deoxyribonucleic acid**
MVC	**model–view–controller**
N	**cardinality of data set**
μ	**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.)

```
partition fcm_possibilistic::doPartition
                              (const dataset & ds)
{
   try
   {
      if (_nClusters < 1)
         throw std::string ("unknown␣number␣of␣clusters");
      if (_nIterations < 1 and _epsilon < 0)
         throw std::string ("You␣should␣set␣a␣maximal␣number␣of␣iteration␣or␣minimal␣difference␣--␣epsilon.");
      if (_nIterations > 0 and _epsilon > 0)
         throw std::string ("Both␣number␣of␣iterations␣and␣minimal␣epsilon␣set␣--␣you␣should␣set␣either␣number␣of␣iterations␣or␣minimal␣epsilon.");

      auto mX = ds.getMatrix();
      std::size_t nAttr = ds.getNumberOfAttributes();
      std::size_t nX    = ds.getNumberOfData();
      std::vector<std::vector<double>> mV;
     mU = std::vector<std::vector<double>> (_nClusters
      );
    for (auto & u : mU)
        u = std::vector<double> (nX);
    randomise(mU);
    normaliseByColumns(mU);
    calculateEtas(_nClusters, nX, ds);
    if (_nIterations > 0)
    {
        for (int iter = 0; iter < _nIterations; iter++)
        {
            mV = calculateClusterCentres(mU, mX);
            mU = modifyPartitionMatrix (mV, mX);
        }
    }
    else if (_epsilon > 0)
    {
        double frob;
        do
        {
            mV = calculateClusterCentres(mU, mX);
             auto mUnew = modifyPartitionMatrix (mV, mX);

             frob = Frobenius_norm_of_difference (mU, mUnew);
            mU = mUnew;
        } while (frob > _epsilon);
    }
   mV = calculateClusterCentres(mU, mX);
    std::vector<std::vector<double>> mS = calculateClusterFuzzification(mU, mV, mX);

```

```
        partition part;
        for (int c = 0; c < _nClusters; c++)
        {
            cluster cl;
            for (std::size_t a = 0; a < nAttr; a++)
            {
                descriptor_gaussian d (mV[c][a], mS[c][a]);
                cl.addDescriptor(d);
            }
            part.addCluster(cl);
        }
        return part;
    }
    catch (my_exception & ex)
    {
        throw my_exception (__FILE__, __FUNCTION__,
            __LINE__, ex.what());
    }
    catch (std::exception & ex)
    {
        throw my_exceptionn (__FILE__, __FUNCTION__,
            __LINE__, ex.what());
    }
    catch (std::string & ex)
    {
        throw my_exception (__FILE__, __FUNCTION__,
            __LINE__, ex);
    }
    catch (...)
    {
        throw my_exception (__FILE__, __FUNCTION__,
            __LINE__, "unknown␣expection");
    }
```

```
}
```

CD/DVD content

The thesis is accompanied by a CD containing:

- **thesis (LaTeX source files and final `pdf` file),**
- **source code of the application,**
- **test data.**

List of Figures

List of Tables