Insert here your thesis' task.

Czech Technical University in Prague
Faculty of Information Technology
Department of Theoretical Computer Science

Master's thesis

Meeting Scheduler

Bc. Lukáš Koc

Supervisor: Ing. Jan Baier

November 27, 2016

Acknowledgements

THANKS (remove entirely in case you do not with to thank anyone)

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on November 27, 2016 ....

Citation of this thesis

Koc, Lukáš. *Meeting Scheduler.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2016.

Abstrakt

V několika větách shrňte obsah a přínos této práce v českém jazyce.

Klíčová slova Replace with comma-separated list of keywords in Czech.

Abstract

Summarize the contents and contribution of your work in a few sentences in English language.

Keywords Replace with comma-separated list of keywords in English.

Contents

List of Figures

Introduction

Chapter 1

State-of-the-art

1.1 Geographic information system

Geographic information systems (GIS) are solving problems which are based on geospatial information. To achieve the goal special tools are being used such as remote sensing tools, geography tools and visualization software. Remote sensing tools gain information on specific object or area from a distance. Geography tools help to observe and research the environmental changes of Earth and its resources, evolution of society or species. Visualization software then displays gathered data as 2D or 3D images [1].

With a drastic change of modern technologies and enormous amount of data a new science was born—geographic information science—which is focusing on geographic concepts, applications and systems. The new science opens door to new problems and issues at global scale, not easily imaginable a few years ago.

The thesis is using knowledge gathered through the course of time in geographic information science to map graph theory problems on a real life data. The developed application belongs to the software category GIS software.

1.2 A need for data

As stated in the bachelor thesis [2], proficient functioning of the application requires fitting data source. The application needs to work with reliable and (preferably) daily updated data. The area of coverage should be big enough to make the application useful, so that many users would find it convenient. The data format should be unified in order to make manipulation and management effective. Choosing correct data format also enables the application to combine different data sources.

To sum up the data are required to follow certain criteria:

- up-to-date

- verified
- human and computer readable
- easy-to-use and unified format
- freely available
- maintained

1.3 Possible data-sources

While searching for data the focus was centred on sources providing free data of Europe available for general public. Further subsections describe sources which matched the criteria mentioned in section 1.2.

1.3.1 EuroGeographics

EuroGeographics is the membership association consisted of 60 organizations and 46 countries. It was created in year 2002, when the Comitée Européen des Responsables de la Cartographie Officielle (CERCO) and the Multi-purpose European Ground Related Information Network (MEGRIN) merged together. Its goal is to gather and collect spatial and infrastructural data of Europe [3].

EuroGeographics association provides following products: EuroBoundaryMap, EuroGlobalMap, EuroRegionalMap and EuroDEM. Boundary map mostly covers borders and administrative informations, DEM map is commonly used for environmental change research or hydrologic modelling. EuroGlobalMap and EuroRegionalMap consists of many datasets: the administrative boundaries, the water network, the transport network etc. In order to download the data it is required to fill up the registration form.

EuroGeographics provides data in following formats

- Geodatabase
- Shapefile

1.3.2 OpenStreetMaps

OpenStreetMaps (OSM) is a project officially supported by the OSM Foundation. OSM was created to build and provide open[1] geographical data available to everyone.

The OMS project was inspired by Wikipedia and is working exactly the same: Users are the ones contributing with their maps, gps measurements,

[1] Open data means for any purpose as long as the OSM and it's contributors are credited.

aerial photographs etc. Since OSM creation in 2004 its community has significantly increased and the data are being updated daily. OSM provides data in their .osm format, which follows XML rules.

In course of time a lot of project was created which works with OSM maps. Thanks to the team Mapzen and their Metro Extract project it is possible to download any major city data in additional two GIS data formats:

- Geojson
- Shapefile

1.3.3 EEA

The European environmental agency (EEA) is agency of European Union providing information about environment for general public. According to their official site [4] it currently consists of 33 member countries.

EEA offers plenty different datasets, maps and graph about nationally designated areas, ecosystem types of Europe, water state and quality, national communications etc.

Depending on the type, these data are provided in following formats:

- Excell table
- CSV
- Shapefile

Most of the datasets are displayed in interactive maps available on the EEA website.

1.3.4 European Observation Network for Territorial Development and Cohesion

The European Observation Network for Territorial Development and Cohesion (ESPON) 2013 Programme is mainly financed from European Regional Development Fund (ERDF) and the main goal is:

"Support policy development in relation to the aim of territorial cohesion and a harmonious development of the European territory ... " [5]

Data are available as soon as user registers and accepts the Term & Conditions. EPSON 2013 data are handled according to ISO 19115 scheme in two formats:

- XML
- Excel file

1.4 Data format

Geographical data exists in many different formats depending on the type and usage of the data. Data representing the elevation of the mountains are better stored in different format than the data representing the location of points of interest. Most of the existing formats typically falls into two main categories: vector formats and raster formats.

Both of them offer two different ways how to represent spatial data. However, the differences between vector and raster data types are equivalent to those in the graphic design world. For better understanding serves the picture 1.1 where is graphically explained how these two types differ.

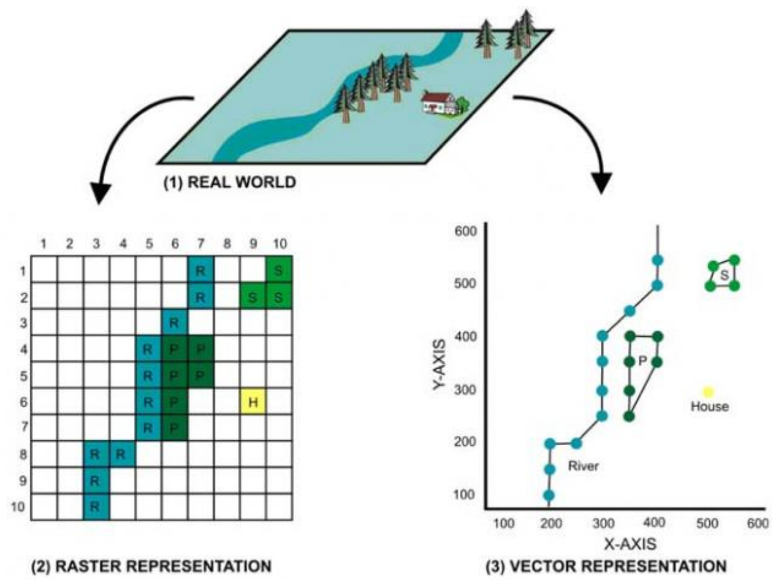

Figure 1.1: Difference between raster and vector data representation

Both representations carry various set of advantages and disadvantages. These will be described in the following subsections.

1.4.1 Raster representation

Raster type formats consist of equally sized cells arranged in rows and columns to construct the representation of space. Individual cell contains an attribute value and location coordinates. Together they create images of point, line, area, network or surface.

Advantages

- Easy and ”cheap” to render

- Represent well both discrete (urban areas, soil types) and continuous data (elevation)
- Grid nature provides suitability for mathematical modeling or quantitative analysis

Disadvantages

- Large amount of data
- Scaling required between layers
- Possible information loss due to generalization (static cell size)
- Difficult to establish network linkage

1.4.2 Vector representation

Vector type formats uses vertices as a basic unit. Vertex consist of x and y coordinates to determine it's position. Using vertices it is possible to create any shape to describe any object. One vertex create point, two can create line etc. Object created by vertices may contain additional attributes describing the feature.

Advantages

- Topology nature
- Compact data structure
- Easy to maintain
- Bigger analysis capability

Disadvantages

- For effective analysis static topology needs to be created
- Every update rebuilding topology is necessary
- Continuous data not effectively represented

1.5 Summary – data usage

Most of the official European sources (EuroGeographics, EPSON) require filling a form for each download of data. This process can be bottom neck for maintenance part of the application while updating the data would take big amount of time and required additional functionality.

Chapter 2

Analysis and design

2.1 Data storage

Since the data source and format question is resolved, next step is to decide the representation of graph in the memory. During the history of graph theory there are three main representations to choose from. In following subsections we will take closer look at all three options.

2.1.1 Adjacency list

Adjacency list stores graph as a list of vertices. Each vertex then contains an information about it's adjacent vertices in form of linked list. Adjacency list is easy to implement and use. All vertices in a graph are mapped onto the array of pointers referencing to a first node of a linked list. In case a vertex does not have the adjacent vertices its pointer is set to null. An example of adjacency list for a simple graph can be found in Figure 2.1.

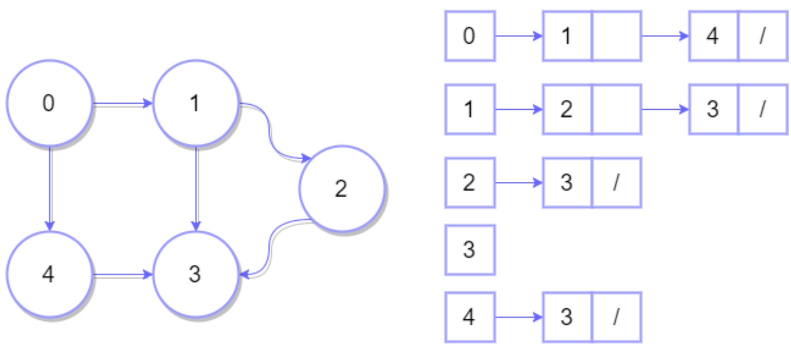

Figure 2.1: Representation of graph using adjacency list

2.1.2 Adjacency matrix

Adjacency matrix is defined as matrix of a size $|V(G)| \times |V(G)|$, where $V(G)$ is a set of all vertices in graph G. Values within the matrix depends on the type of the graph. Generally for unweighted graph we define adjacency matrix as a $\mathbf{A}(G) = [a_{ij}]$, where a_{ij} is the number of edges joining v_i and v_j. If the graph is weighted, the values are from interval $\langle 0, \infty)$, where 0 means two vertices are not adjacent and any non-zero value means they are adjacent with an edge cost of that value[6]. Although between every two vertices must exist 1 edge at most. For the graph G in the Figure 2.1 adjacency matrix $\mathbf{A}$ looks exactly like:

$$\mathbf{A}(G) = \begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccc} v_0 & v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Rows and columns represent vertices of a graph. In my case first row and first column represents vertex 0, second row and column vertex 1 etc. Value in third row and fourth column means that vertex 2 is adjacent with vertex 3. It is noticeable that because my example graph is directed, adjacency matrix is not symmetric.

2.1.3 Incidence matrix

Incidence matrix is very similar to adjacency matrix, but instead of showing relations between vertices themselves it represents relation between vertices and edges. Which means size of incidence matrix is $|V(G)| \times |E(G)|$, where $V(G)$ is a set of all vertices and $E(G)$ is set of all edges in graph G. Incidence matrix of graph G is then $\mathbf{M}(G) = [m_{ij}]$, where m_{ij} is the number of times (0, 1 or 2 in case of loop) that vertex v_i and edge e_j are incident[6].

Interesting case is incidence matrix for directed graph. In that case the sign of the value within matrix $\mathbf{M}$ describes the orientation of the edge. Given the edge $e = (x, y)$ then in the row of vertex x and corresponding column for edge e is value is positive and in the row of vertex y and corresponding column for edge e is negative. For the graph G in the Figure 2.1 incidence matrix $\mathbf{M}$ looks exactly like:

$$\mathbf{M}(G) = \begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccccc} e_0 & e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

2.1.4 Sparse matrix

In mathematics matrices can be divided into two groups: sparse matrices and dense matrices. The definition might sound somehow vague, but sparse matrix is matrix containing huge amount of zero elements. Dense matrix is the exact opposite: containing very few zero elements. In previous subsections it is noticeable that each of the matrices (adjacency and incidence) consist of a lot of zero elements and only few are actually useful values.

The amount of non-zero elements in the adjacency or incidence matrix depends purely on degree of vertices in the graph. In both types of matrices, each row serve as a vertex and within non-zero values represent edges incident to the vertex. Depending on if the graph is directed or undirected the amount of non-zero elements in adjacency matrix will differ. Incidence matrix does not change its number, because it differs only in sign of the value.

For undirected graphs the number of non-zero elements equals to

$$\sum_{v \in V} \mathrm{deg(v)} = 2|E|$$

where E set of all edges and V set of all vertices in graph. Same applies to the directed graph for incidence matrix. For directed graph and adjacency matrix we can observe that the number of non-zero elements depends on amount of the outgoing edges $\Rightarrow$ out-degree, which is

$$\sum_{v \in V} \mathrm{deg}^{-}\mathrm{(v)} = |E|$$

where E is set of all edges, V is set of all vertices in graph and $deg^{-}(v)$ function returns number of outgoing edges from the vertex v.

The reason why I mentioned sparse matrix in the first place is that there are functions and operations which could be done above the sparse matrices. The main motivation for this section are the storage schemes in which sparse matrix could be stored. Using storage scheme enables all the advantages of the regular matrix representation with significantly less memory usage since only the non-zero elements are being stored.

According to [7] there are X storage schemes.

Coordinate format belongs to the simplest storage schemes of sparse matrices. The data structure consists of three arrays: an array containing all the (real or complex) values of the non-zero elements of the original matrix in any order, an integer array containing their row indices and a second integer array containing their column indices. All three arrays are of length N, which is the number of non-zero elements.

Let us take a look at adjacency **A** matrix from section 2.1.2. Clearly this matrix contains less non-zero elements, therefore it is an example of sparse matrix. Using coordinate format matrix **A** looks the following way:

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \quad \begin{array}{r l} AA: & \boxed{1 \mid 1 \mid 1 \mid 1 \mid 1 \mid 1} \\[1em] IR: & \boxed{0 \mid 1 \mid 2 \mid 0 \mid 1 \mid 4} \\[1em] IC: & \boxed{4 \mid 2 \mid 3 \mid 1 \mid 3 \mid 3} \end{array}$$

Array AA stores values of non-zero elements, array IR stores the row index of the corresponding element and array IC stores the column index of the corresponding element. The memory needed for storing matrix is now only $3N$ instead of the original N^2.

If the elements inside array AA are listed by row, the array IR could be transformed to store only indices of the beginning of each row instead. The size of newly defined array IR is then $n+1$, where n is number of rows in original matrix. On the last position (+1) is being written the number on non-zero elements within the original matrix. It also may be represented as address, where fictional $n+1$ row begins.

Array **A** then would be by this scheme described in the following way:

$$\begin{array}{r l} AA: & \boxed{1 \mid 1 \mid 1 \mid 1 \mid 1 \mid 1} \\[1em] IR: & \boxed{0 \mid 2 \mid 4 \mid 5 \mid 5 \mid 6} \\[1em] IC: & \boxed{1 \mid 4 \mid 2 \mid 3 \mid 3 \mid 3} \end{array}$$

The transformation of the IR array and listing elements inside AA by row is called Compressed Sparse Row (CSR) format. In scientific computing CSR format is most commonly used for vector-matrix multiplication while having low memory usage. On the other hand coordinate format excels with itś simplicity and flexibility. Compressed Sparse Row format through the years develops in many number of variations. While storing columns instead of rows, we create a new scheme known as Compressed Sparse Column (CSC) format.

The last scheme I would like to point out is the Ellpack-Itpack format which is very popular on vector machines. The Ellpack-Itpack format stores matrix in two 2-dimensional arrays of the same size $n \times N_{mpr}$, where n is the number of rows of the original matrix and N_{mpr} represents maximum of non-zero elements per row. The first array contains non-zero elements of original matrix. If the number of non-zero elements is less then the N_{mpr}, the rest of the row is filled with zeroes. The second array stores the information about the column in which specific non-zero element is located. For each zero in the first array can be added any number. The Ellpack-Itpack format for the matrix TODO is TODO.

2.1.5 List and matrix comparison

2.2 Definitions and problem description

2.2.1 Directed graph

A directed simple *graph* G is a pair (V, E), where V is a finite set of *vertices* and $E \subseteq V \times V$ are the *edges* of a graph G. The number of vertices $|V|$ is denoted by N and the number of edges $|E|$ is denoted by m throughout this thesis, . A *path* in G is a sequence of vertices $v_1, v_2 \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$ for all $1 \leq i < k$. A path with $v_1 = v_k$ is called a *cycle*. A graph (without multiple edges) can have up to n^2 edges.

2.2.2 Shortest path problem

Let $G = (V, E)$ be a directed graph whose edges are weighted by a function $f : E \rightarrow \mathbb{R}$. The length of a path is the sum of the weights of its edges. In this sense the weights can be reinterpreted as a edge lengths. A cycle whose edges sum to a negative value is *negative cycle*.

The shortest-path problem consists in finding a path of minimum length from a given source $s \in V$ to a given target $t \in V$. Note that the problem is only well defined for all pairs, if G does not contain negative cycles. Since our problem is based on the real world values (distance between two points) negative weights case does not occur it the rest of the thesis. And even if there are negative weights, but not negative cycles, it is possible, using Johnson's algorithm [8], to convert in $O(n_m + n^2 \log n)$ time the original edge weights $f : A \rightarrow \mathbb{R}$ to non-negative arc weights $f' : A \rightarrow \mathbb{R}_0^+$ that result in the same shortest paths.

For solving shortest path problems exist nowadays many algorithms. Most of them evolved from their predecessors. Each of them solves the problem with different parameters. Following list contains the essential shortest path problem algorithms, which provided solid ground in graph theory science:

- Dijkstra's algorithm
- Bellman-Ford algorithm
- Floyd-Warshall algorithm
- Johnson's algorithm

Dijkstra's algorithm [9] and Bellman-Ford algorithm [10, 11] solve single-source shortest path (SSSP) problem. SSSP problem can be defined as: Given a directed graph $G = (V, E)$, with non-negative costs on each edge, and a selected source node $v \in V, \forall w \in V$, find the cost of the least cost path from v to w. The cost of a path is simply the sum of the costs on the edges traversed by the path. Dijkstra's algorithm is greedy algorithm working with the graph were negative edges are not allowed. Bellman-Ford algorithm is non-greedy version of Dijkstra's algorithm which allows it to work with the graph having negative edges.

Floyd-Warshall algorithm [12, 13] and Johnson's algorithm [8] solve the all-pair shortest path (APSP) problem. Floyd-Warshall algorithm iterates all vertices v in order to find better path for every pair going through v in time $O(N^3)$. Johnson's algorithm first converts all the negative edges into positive one and then applies Dijsktra's algorithm on every node within the graph. For sparse graphs the Johnson's algorithm provides better times than Floyd-Warshall algorithm [14].

2.2.3 Dijkstra algorithm

2.3 User interface design

Chapter 3

Realisation

Conclusion

Bibliography

[1] Chen, J. X. Geographic Information Systems. *Computing in Science & Engineering*, Jan.–Feb. 2010: pp. 8–9, ISSN 1521-9615.

[2] Koc, L. *Plánovač setkání*. Bachelor's thesis, Czech Technical University in Prague, Faculty of Information Technology, 2014.

[3] EuroGeographics. About Us. `http://www.eurogeographics.org/about`, [cit. 2016-09-04].

[4] EEA. Who we are. `http://www.eea.europa.eu/about-us`, [cit. 2016-09-06].

[5] ESPON. ESPON 2013 PROGRAMME. `http://www.espon.eu`, [cit. 2016-09-04].

[6] Bondy, J. A.; Murty, U. S. R. *Graph theory with applications*, chapter The incidence and adjacency matrices. New York: Elsevier Science Publishing Co., Inc., 1976, ISBN 0-444-19451-7, p. 7.

[7] Saad, Y. *Iterative Methods for Sparse Linear Systems*. Society for Industrial Mathematics, 2003.

[8] Johnson, D. B. Efficient algorithms for shortest paths in sparse networks. *Journal of the Association for Computing Machinery*, volume 24, 1977: pp. 1–13.

[9] Dijkstra, E. W. *Numerische Mathematik 1*, chapter A Note on Two Problems in Connexion with Graphs. Mathematisch Centrum, Amsterdam, 1959, pp. 269–271.

[10] Bellman, R. E. On a Routing Problem. *The Quarterly of Applied Mathematics*, volume 16, 1958: pp. 87–90.

[11] Ford, L. R.; Fulkerson, D. R. *Flows in Networks.* Princeton, NJ: Princeton University Press, 1962.

[12] Warshall, S. A theorem on Boolean matrices. *Journal of the ACM*, volume 9, no. 1, January 1962: pp. 11–12.

[13] Floyd, R. W. Algorithm 97: Shortest Path. *Communications of the ACM*, volume 5, June 1956: p. 345.

[14] Cormen, T. H.; Leiserson, C. E.; et al. *Introduction to Algorithms.* MIT Press and McGraw-Hill, second edition edition, 2001, 636–640 pp.

APPENDIX **A**

Acronyms

APSP All-pair shortest path

CERCO Comitée Européen des Responsables de la Cartographie Officielle

GUI Graphical user interface

DEM Digital Elevation Model

ERDF European Regional Development Fund

ESPON European Observation Network for Territorial Development and Cohesion

GIS Geographic information systems

MEGRIN Multi-purpose European Ground Related Information Network

OSM OpenStreetMaps

SSSP Single-source shortest path

XML Extensible markup language

Appendix **B**

Contents of enclosed CD

- `readme.txt` .... the file with CD contents description
- `exe` .... the directory with executables
- `src` .... the directory of source codes
  - `wbdcm` .... implementation sources
  - `thesis` .... the directory of LaTeX source codes of the thesis
- `text` .... the thesis text directory
  - `thesis.pdf` .... the thesis text in PDF format
  - `thesis.ps` .... the thesis text in PS format