Matrix Representation of Graphs

Group 3

Eva Rott
Michael Glatzhofer
Dominik Mocher
Julian Wolf

Institute for Information Systems and Computer Media (IICM),
Graz University of Technology
A-8010 Graz, Austria

15 Mai 2017

Abstract

By representing graphs in a matrix adjancy matrix it is possible to observe special patterns and reveal dependencies which might not be seen in the graph per se. By combining the benefits of both the matraix representation and the graph it self, a very powerfull approach of graph analysing may be achieved.

In this survey we present some powerfull techniques applyable to adjancy matrices to analyze graphs. furthermore, we present some tools utilizing these techniques.

Contents

List of Figures

List of Tables

stlistingnameList of Listings

Listings

Chapter 1

Basics

This chapter explains the different types of graphs and their corresponding matrices, which techniques are basically used on matrices and how to interpret the resulting patterns

1.1 Definitions

Basically, a graph in mathematics is an ordered pair $G = (V, E)$ containing a set of nodes V and a set of edges e. However, some literature refer to nodes as “vertices" (thus the V) or “points". Edges may be called “arc" or lines. On the other hand, in the case of an directed graph, edges may also be called arrows. Moreover:

- V is not allowed to be empty
- E is allowed to be empty
- The **order** of a graph is the number of vertices $|V|$
- The **size** of a graph is the number of edges $|E|$

In this paper, we define that every node in the graph has its distinctive unique id, which never changes. This holds for the reordering of the matrices too - when reordering rows and columns, the corresponding index stays with the column, otherwise the graph would be changed with this operation.

1.2 Types of Graphs

We distinguish between two types of edges (directed and undirected) and two types of cost calculations (weighted and unweighted) between the nodes, which leads to 4 different graphs

1.2.1 directed/undirected

As can be seen in figure **??** undirected edges may be traversed in any direction, whereas directed edges may just be traversed in one direction. In this paper we neither talk about the mathematical **quiver**, a directed graph which has multiple arrows pointing from node x to y, nor about a **multigraph**, which is a graph which contains multiple undirected edges connecting just two nodes.

1.2.2 weighted/unweighted

A graph without weights has uniform costs on all edges, which leads to equal costs by traversing the edges in any order. When adding weights or costs to the edges, this fact changes. The order or selection of traversing the edges does matter as it creates different costs.

TODO: some more stuff here

1.3 Use cases

Some use cases of the different graph types are (to name just a few examples):

- Navigation system (weighted directed)
  - Nodes: Cities/POIs
  - Edges: Routes directed (one way streets)
    - weights
      - length of street (find shortest way)
      - time to traverse the street (find fastest way)
- Subway map (undirected unweighted)
  - Nodes: stations
  - edges: connection between stations
- Relations of tweets (directed unweighted)
  - nodes: single tweet entry
  - edges: references to other tweets

1.4 Matrix representation of graphs

When representing graphs in a matrix, an adjacency matrix is used. Adjacency matrices are structured with every row and every column represents one node. This leads to a N x N square matrix, where N is the number of nodes.

These matrices show some patterns according to their corresponding graph but most times these patterns are not immediately visible. There are some techniques to reveal these patterns, all of them involving the reordering of the matrix.

1.4.1 Reordering

The main goal of reordering the matrix is to cluster the edges and thus reveal certain patterns. An example of this behaviour can be seen in figure 1.1.

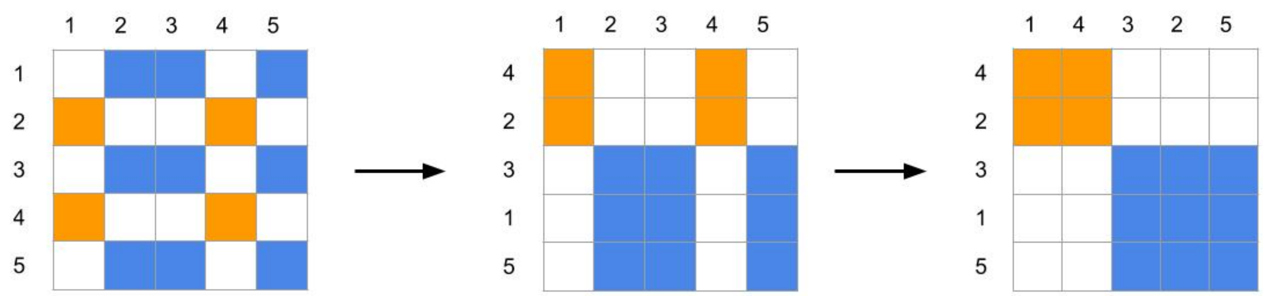

Figure 1.1: Reordering a matrix

When reordering the matrix, the indices of the single rows and columns stay with the rows, otherwise the graph would change by this workstep. In this example, at first the rows 1 and 4 get swapped and as a second step columns 2 and 4. In this way the full connection pattern of the two subgraphs may be observed.

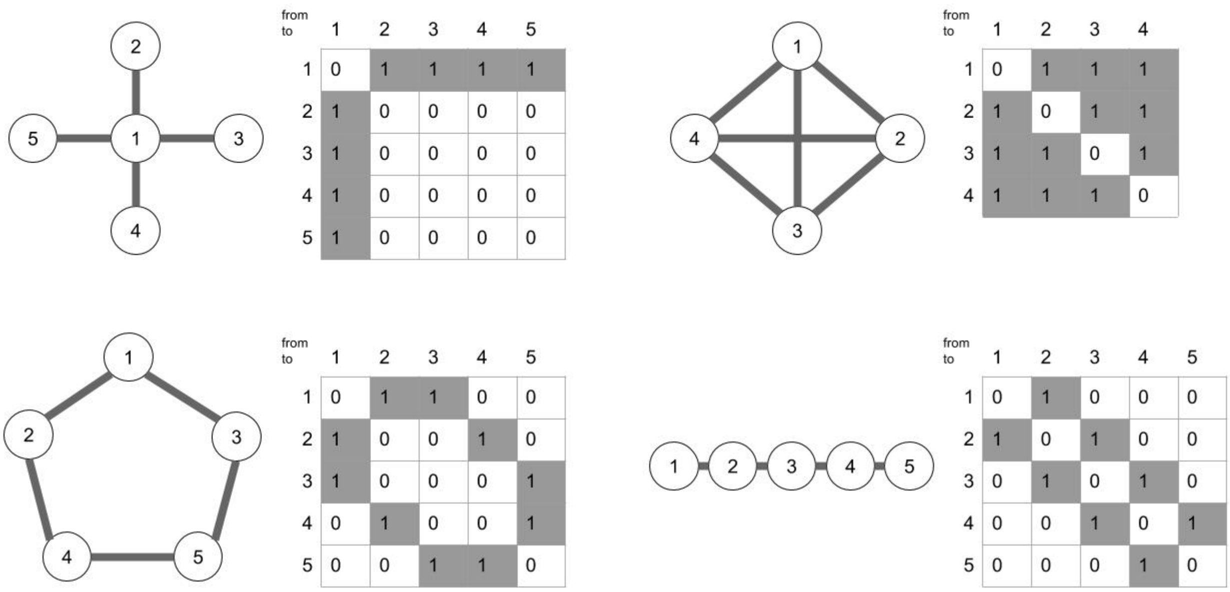

Figure 1.2: 4 differnet patterns: star, full, circle and line

1.4.2 Patterns

There are 4 main patterns which may be revealed by reordering the matrix. These patterns may be combined in such a way, that for example a subgraph creates a circle, but one node if it is connected to every other node. This results in a combination of the star and the circle pattern. The four different patterns can be seen in the corresponding figures 1.2.

Chapter 2

Scalability

Chapter 3

Cell Visualization Techniques

3.1 What to visualize in cells?

Often graph node links contain additional data, besides their connected nodes and weight, for example the textual description.

Most of the time, the cell simply represents the connection between two nodes by filling the cell. If the input is a weighted graph, this information is often extended by the edge weights. Given the case that the input graph is undirected, the cells form a symmetric pattern along the diagonal.

On the other hand, the cell can also represent data of a node, for example the affiliation to a specific cluster or the similarity to nodes from other clusters or the local neighbourhood. Most tools provide the possibility to highlight the current selection in the matrix.

3.2 How to visualize it in cells?

Connections are most of the time shown in a black-white scheme, where black means that there is a connection, and accordingly white shows, that there is none. The current selection is then highlighted by increasing the transparency of the not-selected cells or simply setting them to grey. The logical next step from this is the extension to a wider color scheme, so that for example weights can be represented by different color grades, additionally with text as a fallback. If this scale is discrete, there is also the possibility to display icons or textures instead of color grades. If the data which should be visualized is more exotic, like the similarity of nodes in the local neighbourhood, this could be displayed as bar charts or histograms inside the cells. A matrix cell can even include another matrix and represent the according sub-cells. This technique is mainly used to simplify complex and large matrices.

3.3 Example: Nodetrix

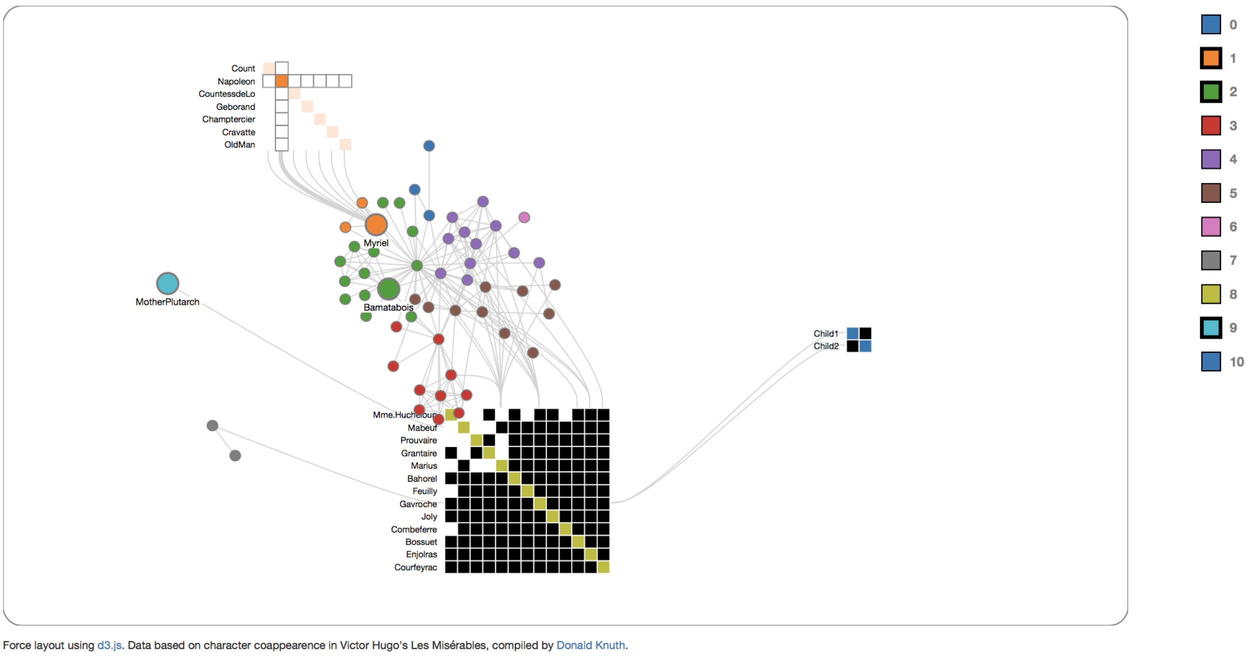

Figure 3.1: TODO: Screenshot created using Nodetrix...

In the matrix representation of Nodetrix, connections are shown as black cells, the color in the matrix diagonal visualizes the affiliation to a specific cluster in the graph. The matrices in Nodetrix have a hover effect, which highlights the row and column of the currently selected cell. Every other cell in the matrix becomes light gray to help the user focus on connections. Additionally, the connections to graph nodes from the selected cell are emphasized too, as they are drawn boldly.

3.4 Example: Matrix Zoom

Matrix zoom uses colors to visualize edge attributes. In the example set, edge attributes are an indication if a call is allowed or not or the local neighbourhood of a call, visualized in a color scale. Therefore, calls, which have a shorter path-distance to the considered call are indicated in red. Transparency is used to indicate the call density of this matrix cell, higher cell density meaning a larger percentage of subcells containing calls. TODO: citation ham-ivis2003.pdf

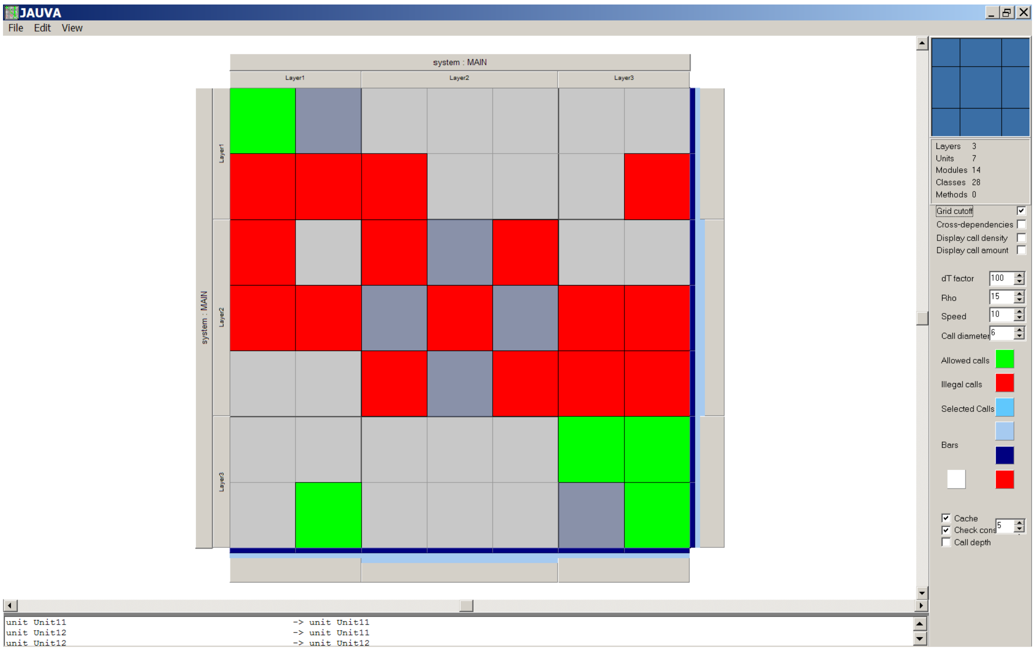

Figure 3.2: Screenshot created using Matrix zoom...

3.5 Example: Cubix

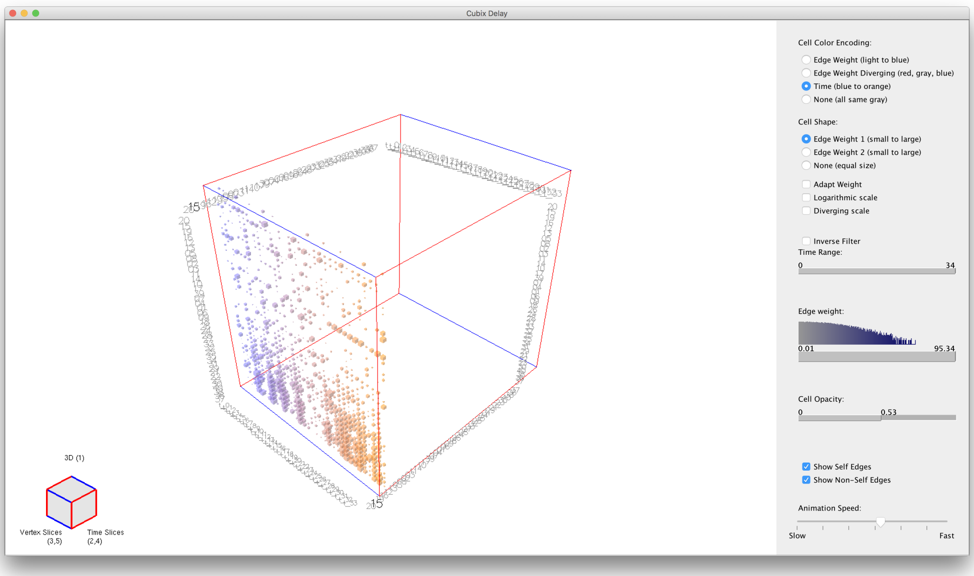

Figure 3.3: The 3D representation of Data in Cubix, screenshot created using Cubix

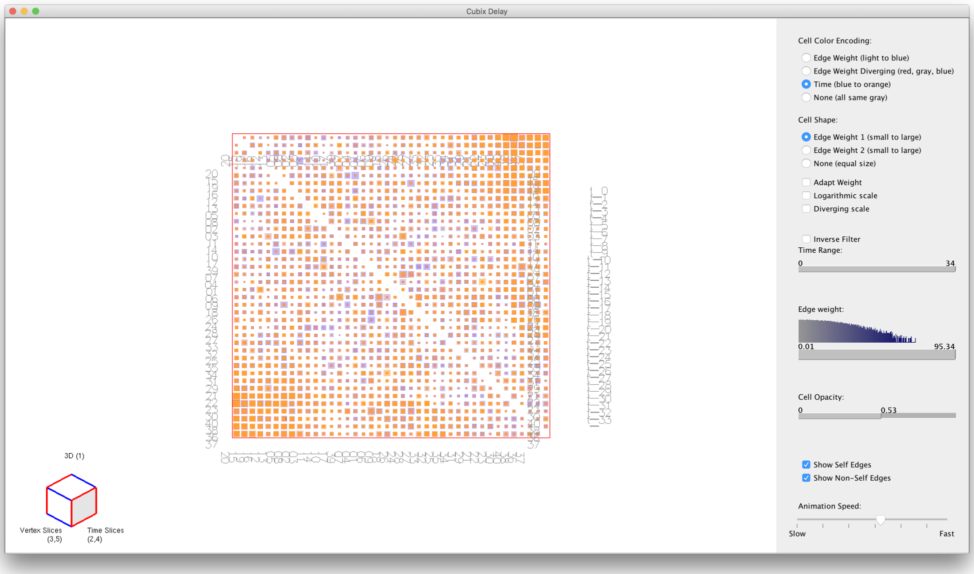

Figure 3.4: The same data in 2D shown as time slices, screenshot created using Cubix

Cubix is a tool to visualize and analyze graphs, which changes over time using the space-time-metaphor. Adjacency matrices are stacked onto each other in chronological order to create a cube with two vertex and one time dimension.

The cell colors represent by default the edge weight, this can also be changed to the edge weight diverging or the affiliation to a specific time slice. The user can switch off the color encoding too, for easier recognizing patterns of the cells. Since this is a three-dimensional visualization, cell opacity is an important method for identifying patterns or cells behind another one.

Additionally, the size of the cell is another tool to visualize the weight of the edges and their change over time. For pattern finding, this feature can also be turned off. TODO: citation Bach2014cubix.pdf (http://www.aviz.fr/ bbach/cubix/Bach2014cubix.pdf)

Chapter 4

Matrix headers