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

List of Listings

Chapter 1

Basics

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

1.1 Definitions

In mathematics a graph 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, 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

Basically, there are two types of edges (directed and undirected) and two types of cost calculations (weighted and unweighted), which leads to 4 different graphs. Figure 1.1 shows these 4 different types of graphs.

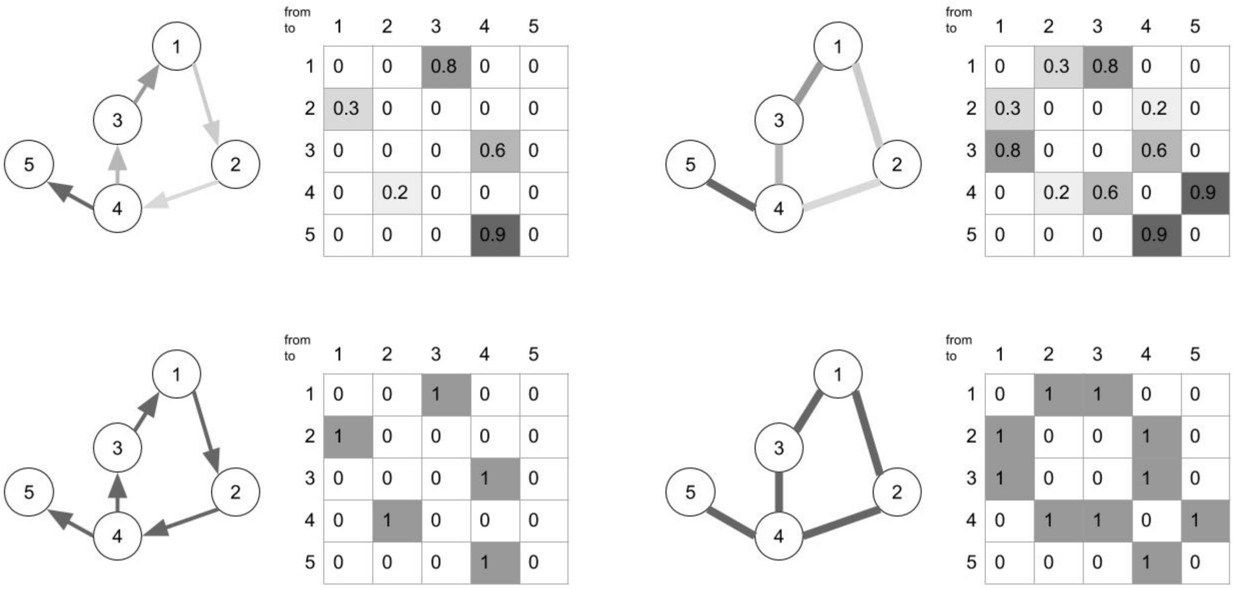

Figure 1.1: 4 different types of graphs (top: weighted directed and undirected, bottom: unweighted directed and undirected)

1.2.1 directed/undirected

Undirected edges may be traversed in any direction, whereas directed edges may just be traversed in one direction. For matrix representation of graphs, neither the mathematical **quiver**, a directed graph which has multiple arrows pointing from node x to y, nor the **multigraph**, which is a graph which contains multiple undirected edges connecting just two nodes, is used.

1.2.2 weighted/unweighted

For graphs without costs defined for their edges, so called **unweighted graphs**, may be processed differently:

- Either the algorithm searches for the shortest path, thus defining a uniform cost on all edges
- Or there is no cost calculation at all, even the amount of edges is ignored

When adding weights to the edges, the graph is called a **weighted graph**. These weights typically represent different things, for example:

- time
- length
- energy consumption
- elevations

to name just a few.

With weights defined on the edges, the approaches of algorithms are different, as the shortest path, in despite of number of traversed edges, is not necessarily the cheapest one.

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 1.2.

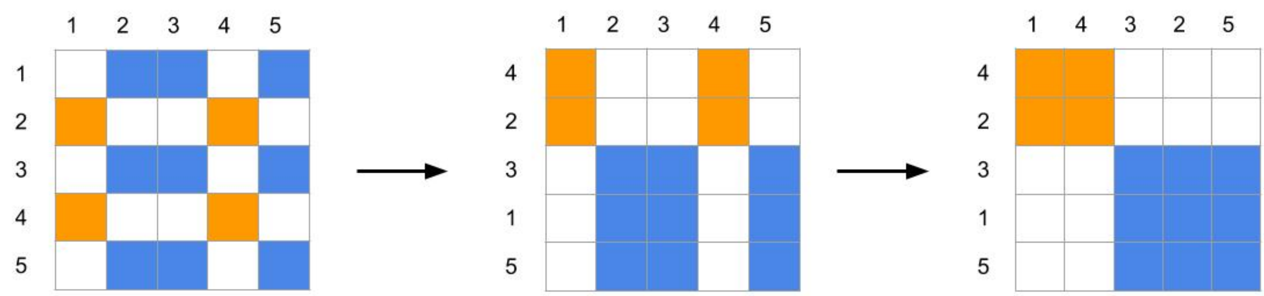

Figure 1.2: Reordering a matrix

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.

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 1.3.

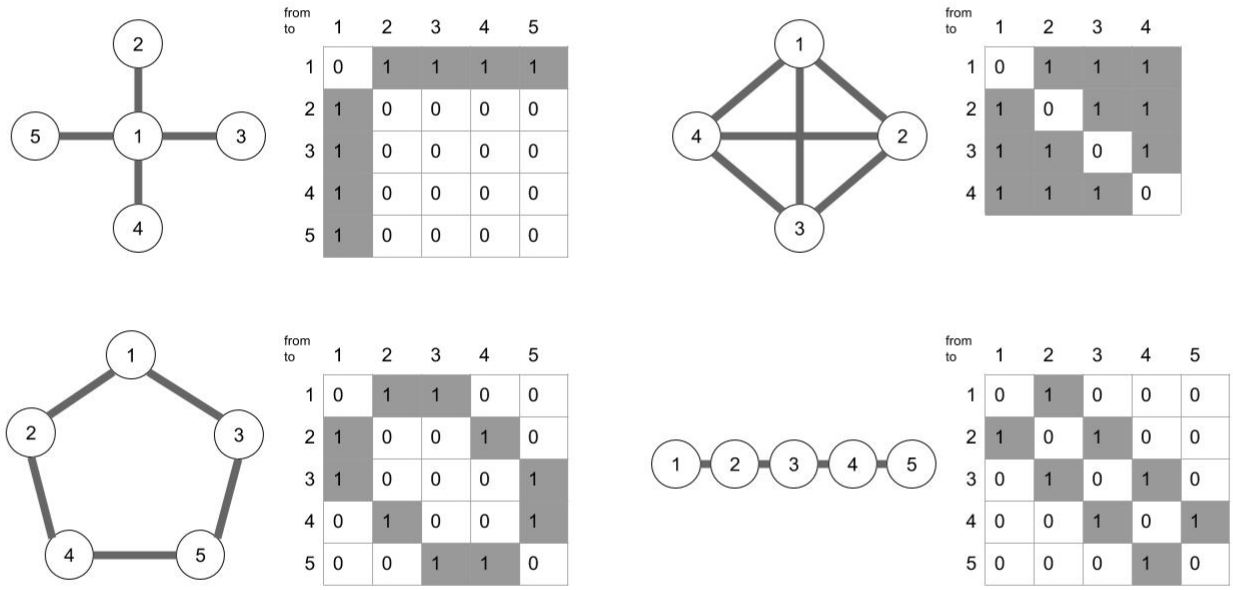

Figure 1.3: 4 differnet patterns: star, full, circle and line

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

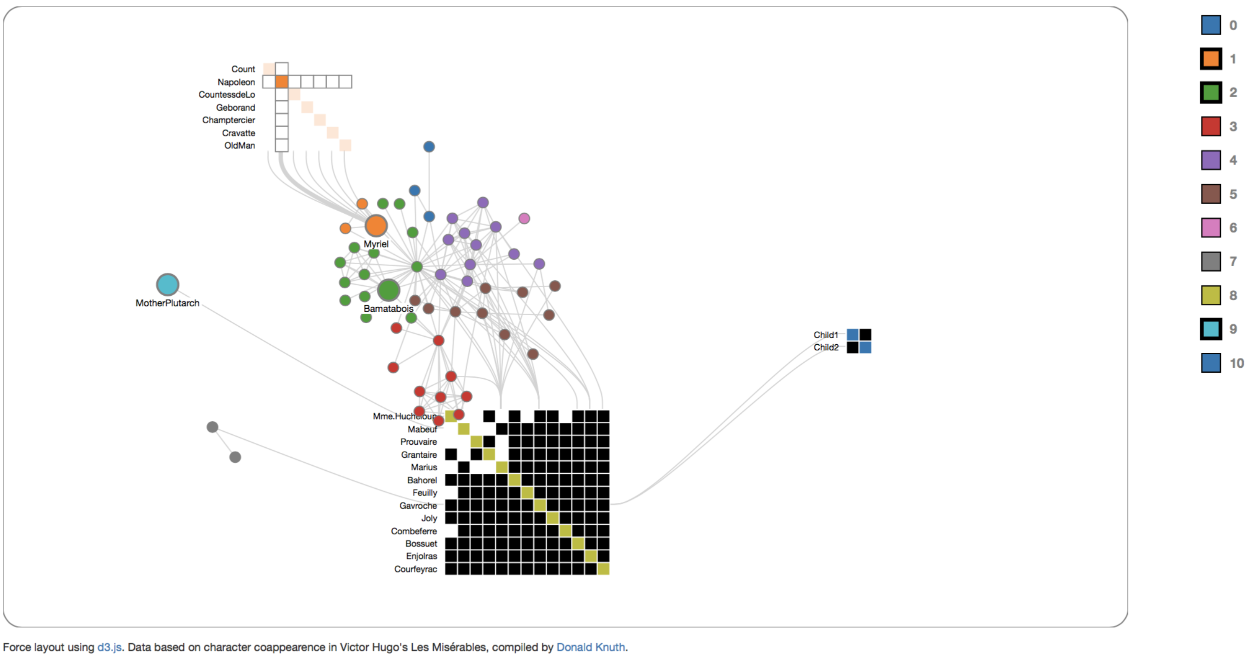

Figure 3.1: The demo application of Nodetrix. Screenshot created using Nodetrix. [Henry et al., 2007, pages 1302-1309].

In the matrix representation of Nodetrix in figure 3.1 can be seen, that connections are shown as black cells and 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. [Henry et al., 2007, pages 1302-1309]

3.4 Example: Matrix Zoom

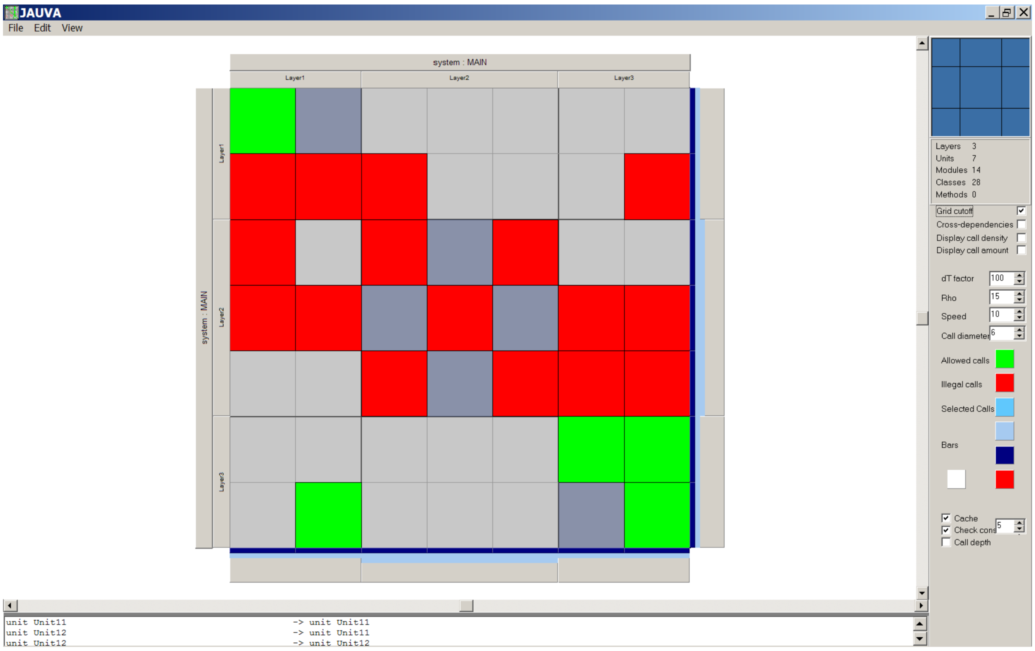

Figure 3.2: The color encoding of Matrix Zoom. Screenshot created using Matrix Zoom. [van Ham, 2003, pages 227–232]

Matrix zoom uses colors to visualize edge attributes, as shown in figure 3.2. 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. [van Ham, 2003, pages 227–232]

3.5 Example: Cubix

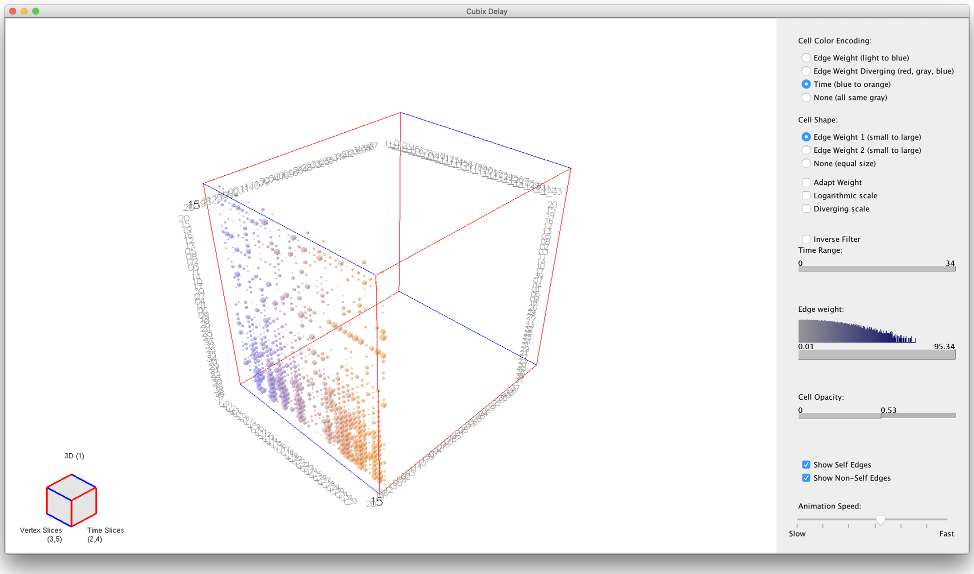

Figure 3.3: The 3D representation of Data in Cubix, screenshot created using Cubix.[Bach et al., 2014, pages 877–886]

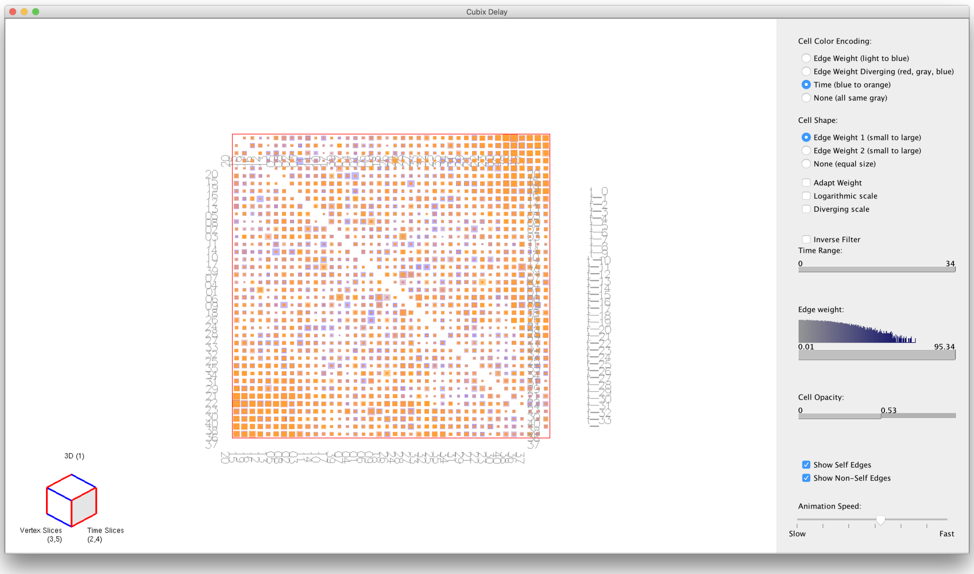

Figure 3.4: The same data in 2D shown as time slices, screenshot created using Cubix. [Bach et al., 2014, pages 877–886]

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, as it is shown in figure 3.3.

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. For further investigation, a single time slice can be inspected as seen in figure 3.4

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. [Bach et al., 2014, pages 877–886]

Chapter 4

Reordering

Reordering describes the process of either moving nodes in a graph or moving rows or columns in a matrix. There are two types of reordering: manual and automatic. Manual reordering is done by the user of a software. This survey focuses on automatic reordering, which is done by a software tool based on its implemented algorithms. The information used as input for the algorithms can be the node label, node in / out degree or clustering data. The mentioned information sources are not a complete list of available reordering input data. However, they were found in the tested programs and they are described in more detail in the following enumeration.

1. Reordering using node label. The data used for this example describes a number of functions of a program connected corresponding to the its control flow. Figure [$function_{c}alls_{g}raph.png$] TODO shows the directed graph with the functions as nodes; Figure [$sample_{m}atrix.png$] TODO the unsorted matrix representation of the graph. The result of reordering the matrix based on alphabetic label name order can be seen in figure [$reordered_{b}y_{l}abel.png$] TODO.

2. Reordering using node out degree. The data used for this example is the same as in TODO. Figure TODO shows a reordered matrix based on ascending order of node in and out degree. The function with the largest number of incoming links is the rightmost column and the node function with the largest number of outgoing links the lowermost row of the matrix.

3. Reordering using node clustering. As node clustering needs to be computed first, this type of reordering is explained in an example taken from the program Nodetrix. Displayed in figure TODO [$nodetrix_{r}eordering.png$] is the graph with the clustered nodes, marked in different colors, and some sub-matrices for some of the ordered clusters.

Chapter 5

Matrix headers

A standard matrix visualization contains a symmetric grid representing the node connections and node labels on top and to the left side of the grid. In this survey this area and in general the area around the grid is referred to as matrix header. The matrix header can be used for visualizing additional information about the matrix data. An example for that is a group of node connections, the node density or the current level of zoom in a multi-layer matrix.

There are various techniques to achieve additional information visualization in a matrix header. Most of the visualization techniques can be found in the program Matrix Explorer. The Matrix Explorer is another tool for matrix visualization of graphs. As seen in figure [*matrixExplorer.png*] TODO, it displays the matrix without a zoom functionality, while giving a small overview of the full matrix in the top left corner of the graphical user interface. Aside from the matrix visualization, the Matrix Explorer provides various options for filtering or sorting of nodes and executing operations on the matrix headers. The following section describes the most common operations and gives one example per listed tool for a better understanding.

1. Lines in the matrix header. As seen in figure [$path_{h}ighlighting.png$] TODO from the program MatLink, created by Henry et. al [$henryphd2008ref!$] curved lines are used for highlighting node connections. The shortest path is highlighted in red. When one node is selected, the program draws the paths in the headers of the matrix. Using this visual information it can quickly be seen how many other nodes are connected directly to the selected node. In addition, the path from one node to another connected node can be traversed using these lines. TODO [$refhenry2008phd$]

2. Histogram per node in the matrix header. A histogram can be computed over various node properties. An example for such a property is the node degree. The Matrix Explorer offers this histogram functionality. Figure [$better_{h}isto.png$] TODO shows a matrix visualization of data similar to that used for the first two examples in the previous chapter. Again, a set of functions from a program are represented as nodes and the program's control flow as links. Looking at the histogram which was computed over the outgoing links it becomes obvious that the main function has the largest light gray area. This means that it has the largest number of outgoing links of all nodes. In contrast to the node out degree distribution, the incoming links follow a more balanced distribution. Considering this example, a histogram in the header is well suited for showing the general distribution of the node degree. In case nodes containing extreme values shoulreorderingd be highlighted, the header color visualization technique can be used.

3. Colors in the matrix header. Every node in a graph can be assigned a color in a certain range. This color distribution assigned to the nodes can be computed for the same properties as the histogram. The darker the color the higher the value of this node. Considering the same example and figure [$better_{c}olor.png$] TODO as in the previous section, it becomes obvious that the start function has no incoming links as it is the root function of the program.

4. Histogram per section in the matrix header. In contrast to the histogram computed per node as explained in [$point2$] TODO, a histogram can also be computed over multiple columns or rows at the same time.

Figure [$matrixzoom_histo.png$] TODO shows an example. The screenshot was taken from the matrix visualization program MatrixZoom. The displayed matrix is divided into three times three sub-sections. For every group of three vertically or horizontally aligned sections the amount of data contained in it is computed. Next, those values are transformed into a histogram representation, which is then drawn as light blue bars on the right and bottom matrix header. The result shows that the center section of the matrix has the largest density of data points of all sub-sections.

Bibliography

Bach, Benjamin, Emmanuel Pietriga and Jean-Daniel Fekete [2014]. “Visualizing Dynamic Networks with Matrix Cubes”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’14. Toronto, Ontario, Canada: ACM, 2014, pages 877–886. ISBN 978-1-4503-2473-1. doi:10.1145/2556288.2557010. http://doi.acm.org/10.1145/2556288.2557010 (cited on pages 10–11).

Henry, Nathalie, Jean-Daniel Fekete and Michael J. McGuffin [2007]. “NodeTrix: a Hybrid Visualization of Social Networks”. *IEEE Transactions on Visualization and Computer Graphics* 13.6 [2007], pages 1302–1309. ISSN 1077-2626. doi:http://doi.ieeecomputersociety.org/10.1109/TVCG.2007.70582 (cited on page 8).

Van Ham, Frank [2003]. “Using Multilevel Call Matrices in Large Software Projects”. In: *9th IEEE Symposium on Information Visualization (InfoVis 2003), 20-21 October 2003, Seattle, WA, USA*. 2003, pages 227–232. doi:10.1109/INFVIS.2003.1249030. http://doi.ieeecomputersociety.org/10.1109/INFVIS.2003.1249030 (cited on page 9).