Creating Dynamic Graphs: A Visualization Tool

Emma van Beek, Sander van Gansewinkel, Jeroen Gommers, Ankit Majhi, Marco van Zelst, Michael Burch
Eindhoven University of Technology

Figure 1: Teaser showcasing dynamics as rendered by the visualization tool on the full data set.

ABSTRACT

This paper describes an interactive tool for the visualization of dynamic graph data sets. This tool allows experts in this field to upload, share, analyse, and compare their data using different forms of dynamic graph visualizations. The current version allows the users to create three different visualizations. Those visualizations can be improved by making use of parameters that are available, depending on the visualization form. The tool is programmed in the Python language, makes use of tkinter, pandas, matplotlib, networkx, and numpy and runs in Jupyter Notebook. The use of the tool is shown by applying it on a real-world data set with four components which can be used to describe a dynamic weighted and directed graph. The output of this data set is given by means of the visualization techniques. The differences, disadvantages, and advantages of the tool compared to other dynamic graph visualization tools will be explained.

Index Terms: Human-centered computing—Visualization—Visualization techniques—; Human-centered computing—Visualization—Visualization design and evaluation methods

1 INTRODUCTION

The visualization of data is important for many data analysts and data scientists, as without it big data would be hard to understand and even harder to use in research [7]. Its visualization makes it possible for the analyst to take a closer look at specific parts of the data and discover relations that were never thought of before.

In this paper, the visualization tool that was created and its evaluation will be introduced. The different visualizations the tool can create will be shown, and the differences the parameters make on these visualizations will be explained. Finally, the tool will be compared to others and the advantages and disadvantages of the tool will be described.

The main goal of the project was to create a visualization tool that is easy to use but still works effectively for the experienced user. Many visualization tools are very complex at first sight, which is something that is avoided in the tool. The tool is very easy to use and to understand. Another requirement was that the tool creates the wanted plots in not too much time without losing the accuracy. This is a trade-off that was considered in the creating of the visualization tool. Also, to create comprehensive graphs, one needs filters for the different dimensions and needs different colour options. The user may want to apply different algorithms on the data set, of which the results must be visible in the graphs. The user also needs to be able to interact with the plot, for example by sliding over with the mouse or clicking on certain parts of the graphs. This last requirement is also implemented in the tool.

The visualization tool currently can create adjacency matrices [3], dynamic graphs [1,6], and node-link diagrams [4].

2 RELATED WORK

There are a lot of graphical user interfaces in the world. For example, all versions of Microsoft Windows are graphical user interfaces [8]. Usually, it is divided in the start button on the bottom left, the taskbar icons next to it, the home button to return to your desktop on the bottom far right, next to that is the time and date area, and lastly, next to that is the windows notification area.

Graphical user interfaces can be used as a desktop environment where their main usage is but there is so much more potential, like altering visualizations with user-based interaction so the user can

decide what to visualize. The focus lies on the latter. The decision to visualize two-dimensional data was made, since this would be less computationally expensive than making 3D visualizations and since the main client can alter the data the visualizations should be understandable which is more likely for a two-dimensional visualization. A related approach is the IMDb explorer [5, 9] which also allows the user to visualize a data set and alter the visualization. But, it does not let the user upload his own data set, which restricts the user.

The tool created will be a solution which does let the user upload his own data, is easy to use, and makes easy-to-interpret visualizations. This would be an addition to the existing works in this domain.

3 Implementation

The group chose to make use of a programming language familiar to all group members, namely Python, within the environment of Jupyter Notebook. The group chose for this language because there are quite a lot of extensive support libraries, which are quite useful in this project. Other benefits of Python include: coding is possible in relatively less steps and it serves a general purpose. Python also possesses features such as interactiveness, dynamic, interpretable, and object-oriented.

For making a graphical user interface, the decision to work with tkinter was made, as this was the most common result coming up when searching for making a GUI in Python [10]. The main libraries used in the project are: numpy, pandas, matplotlib, networkx, and pygraphviz. The components in the software consist of drop-down lists, search fields, buttons, and text fields. The drop-down lists are implemented at the very start, when selecting a data set from files. The search fields are used when choosing a file; by typing a fraction of the file name, the file shows up. The buttons used are for importing a different file, changing the colour of the graph and adjacency matrix and updating the graphs based on time and weight filters. From this it becomes clear that the graph is immediately updating when modifying the values of the components. Running the Python code automatically gets the server running.

4 Data Model

To make visualizations from data, the correct type of data must be uploaded and defects in this data must be cleaned before visualizing.

4.1 Data

The GUI visualizes data sets with numeric values only. Yet, it focuses on hierarchical data. This is data which is organized into a graph structure where the data is stored as records, which is a collection of fields where a field contains only one value. These records are connected through links and thereby it creates a graph. At the top is the root element, the main category, below that the subcategories are formed. To correctly visualize data, sometimes data preprocessing must be done before visualizing the data to prevent data defects being visualized. There are four different aspects of data preprocessing [11]: data cleaning, data integration, data transformation, and data reduction. The chosen aspects to incorporate into the GUI are data cleaning and data reduction.

4.2 Data Cleaning

Data cleaning provides methods to fill up or delete incomplete data and to deal with errors in data. To tackle incomplete data, listwise deletion is used before making the visualization, this means that the entire row is deleted when a missing value is encountered. This is not the best method to cover incomplete data, but it is the least computationally expensive. This can be used because not a lot of data is missing and when it would be missing only a small part of the data set is deleted [12]. Furthermore, there have not been missing values encountered in the used data set. Nothing is done to transform noisy data.

4.3 Data Integration

Data integration problems occur when data is extracted from different sources and these cannot be merged easily. These problems will be prevented if the data carries meta data. When merging two data sets with meta data, attributes are matched and then errors can be prevented. Since there are not data sets merged yet, this is not integrated in the GUI.

4.4 Data Transformation

Data transformation is used to have data in the correct format to use the data. The only data transformation used on this moment is changing the first line of the data set to string and the lines after that to integers. This ensures that numeric values are treated as numeric and that the variable names are really strings.

4.5 Data Reduction

Data reduction is used if huge data sets which take a long running time need to be visualized. Data reduction tries to represent the original data but in a smaller quantity. This is not used since running time can also be reduced in other ways like rewriting the visualization code more efficiently.

5 Visualization Tool

In this section, the details in the visualization tool will be discussed: how it works, what it can do and what it looks like.

5.1 Graphical User Interface

When the user runs the code, a file explorer automatically pops up, in which the user chooses the dynamic graph data set to import. After this the GUI is loading and a dynamic graph visualization is automatically shown. At the left hand side of the GUI, different interactions can be found. The first button is to import another dynamic graph data set and to forget the last one. The user can also choose a different colour for the dynamic graph visualization and the adjacency matrix, choosing between the colour schemes Blues, Greens, Purples, Reds, Yellow-Orange-Red (YlOrRd), Oranges, Greys, and Yellows. The default colour scheme for the plots is the Blues. There are also filters to select the minimum and maximum weight and to select the minimum and maximum time. When these filters are being used, a part of the data is selected. There is also a node filter. The user can enter the wanted nodes in the textbox, comma separated, and only these nodes will be visualized. This filter also shows the nodes that were indirectly selected through edge connections.

At the left bottom, an adjacency matrix is shown. The user has the choice to reorder this matrix. This can be done by clicking on the checkbox and pressing on the Update filters button. This can also be done when other filters have been applied, like selecting weights and timestamps or changing the colour scheme. The user can also run Dijkstra's algorithm on the dynamic graph data set, entering a start and end node and pressing the button for Dijkstra's algorithm. After this, the GUI automatically calculates the shortest path weight and it shows the user what the shortest path is in the dynamic graph visualization. The shortest path weight is shown below the textbox with the start and end node on the left-hand side of the GUI.

Underneath each visualization a row of buttons is placed, these resemble the interactions. The first button is to reset the visualization, the second and third button are the undo and redo buttons. The fourth button is to move the plot around, the fifth button is to zoom in, the sixth button is to change the size of the visualization, and the last one is to save the visualization to a wanted directory on the PC.

5.2 Visualization Techniques

In the GUI, three different visualizations are used: a dynamic graph visualization, an adjacency matrix, and a node-link diagram.

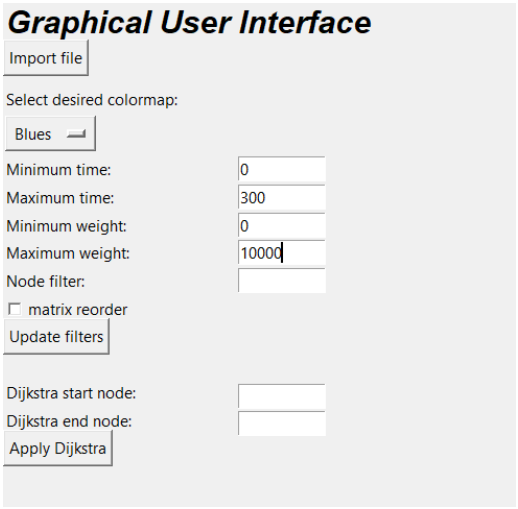

Figure 2: The different parameters from the visualization tool

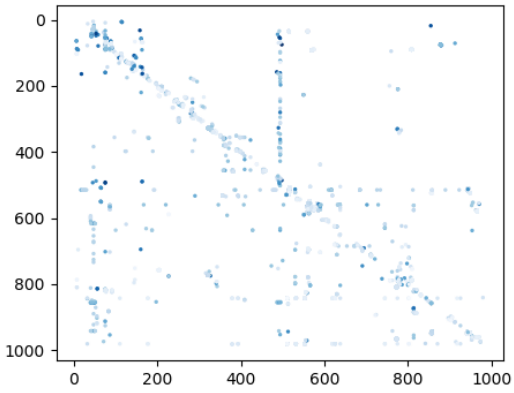

Figure 3: The adjacency matrix from the visualization tool with default parameters

5.2.1 Dynamic Graph Visualization

For every time step in the data set, a bipartite graph visualization is created. These graphs are placed next to each other, making the bipartite graphs interleave in each other. This way, all time steps are shown in one figure with on the x-axis the time and on the y-axis the vertex numbers. Each bipartite graph is 5 time steps wide, meaning for example the first bipartite graph starts at the beginning of the figure and stops at the point where the fifth bipartite graph begins. The weights of the graph are visualized due by colour.

5.2.2 Adjacency Matrix

In this visualization are the edges between the vertices visualized. Every point in this matrix means an edge between the start vertices, which are represented on the x-axis, and the target vertices, which are represented on the y-axis. The weights of the edges are visualized with different colours in the graph. When matrix reordering is been used, the x- and y-axis are not in the original order anymore. This means that the points in the matrix are now on different places than without the reordering. When this happens, all the points in the matrix clutter together in the form of a leaf.

5.2.3 Node-Link Diagram

This visualization can be found at the right bottom corner of the tool. This figure gives a clear overview of the whole data set. All the blue dots represent the nodes (vertices) and all the black lines between the dots represent the links (edges). The user always sees all nodes and their links from the whole data set, independent of the usage of filters. This node-link diagram has been created without any minimal link crossing algorithm and this visualization is not in 3D. In this way the figure becomes unreadable very fast. It is even unreadable for the data set used for the project.

5.3 Interactions

The visualization uses eight different types of interactions, namely: select, explore, encode, filter, reconfigure, abstract/elaborate, undo/redo, and saving. Select is mainly implemented to ensure the possibility of clicking different file names to open data sets. Explore makes it possible to focus on a subset of the chosen data set. The idea behind encoding is showing a different visual representation of the data, which is mainly done by colour coding in the GUI. Different kinds of filters can be applied in the GUI such as weight and time filter, making it possible to scale the data based on the range of weight and time.

For the reconfigure interaction, matrix reordering is used. For all the visualizations it is possible to zoom in. This geometric zooming function is an example for the abstract/elaborate interaction. When the user does not touch the filter buttons and only uses the zoom-in function, the graph can be reset to the original or it can go just one step back or forward, using the home, undo, and redo buttons. The last interaction the GUI has, is the saving interaction. This can be simply done by clicking on the save button. After this the user can save the picture of one graph in some directory with a name.

Figure 4: Interactions in the GUI. Left to right: back to original picture - undo - redo - move plot around - cut out - change dimensions of plot - save plot

Figure 5: x- and y-coordinates of a datapoint in the adjacency matrix, created by mouse interaction

5.4 Algorithms

In this GUI, two different algorithms are used: Dijkstra's algorithm and the matrix reordering algorithm reversed Cuthill McKee.

5.4.1 Dijkstra's Algorithm

Dijkstra's algorithm [13] can search the shortest path between nodes in a graph. Through a set of pre-determined computations, the user gets from point A to the wanted point in the fastest way. This only works when the weights of the vertices are non-negative. The worst case running time of this algorithm is $O(V^2)$, where V is the number of nodes. Since the algorithm has a big running time, it takes a long time to compute the shortest path using Dijkstra's algorithm on a graph with a lot of nodes. However, this algorithm is still chosen since other shortest path algorithms have a higher memory complexity which can lead to an out-of-memory error when running with a laptop on big data.

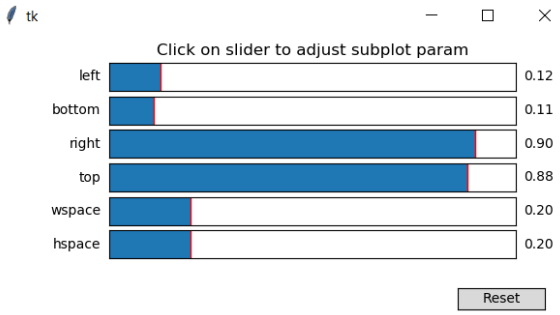

Figure 6: Options for changing dimensions of the plots in the GUI

5.4.2 Reversed Cuthill McKee Algorithm

Reversed Cuthill McKee [14] is a matrix reordering algorithm which changes the adjacency matrix into a band matrix form with a small bandwidth, but the index numbers will reverse. The algorithm runs in polynomial time $O(E+V)$ and it works by creating levels for all nodes. Then a set is created from these levels by listing all vertices adjacent to all nodes. Then these nodes are listed in increasing degree. This is a useful algorithm to use since it is fast and the results are always in the same form so comparisons between different data sets can be made easily.

6 Application Example

To explain how the visualization tool works in practice, the tool will be used to give new insights in the data set provided in the visualization course. Several visualizations made with the tool will be displayed.

When running the code in Jupyter Notebook, the GUI will be built. Unfortunately, the code takes quite a while to run. Firstly, the code asks the users to upload a data set, which can be selected the same way one would normally upload or open a file at their PC.

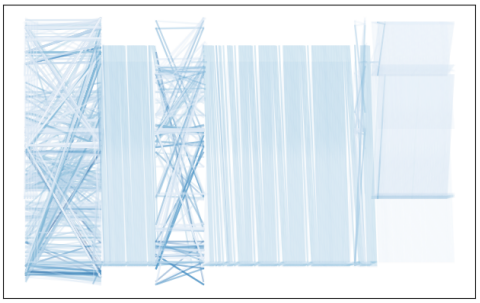

Figure 7: The dynamic graph from the visualization tool with default parameters

After the data set is selected, the GUI will be further built, and the dynamic graph is created. Since the default colour map of the graphs in the tool is the Blues, both the dynamic graph and the adjacency matrix will have the blue colour transitions.

The GUI also immediately creates a node-link diagram. This is visible in the left bottom corner. Unfortunately, this diagram does not change when the parameters are changed, like the dynamic graph and the adjacency matrix do, but the user can interact with the plot. Its colour is blue, since that matches the default setting for the colour scheme of the other plots. It is also made under the default parameters for the other two plots.

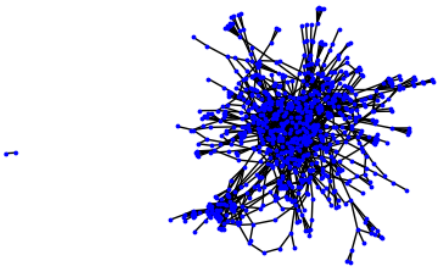

Figure 8: The node-link diagram from the visualization tool

After the basic graph has been displayed in the visualization tool, filters can be used on the data set. For example, a graph with the weights between 0 and 100 instead of the default between 0 and 10,000 can be made. Selecting these gives the following dynamic graph in Figure 9:

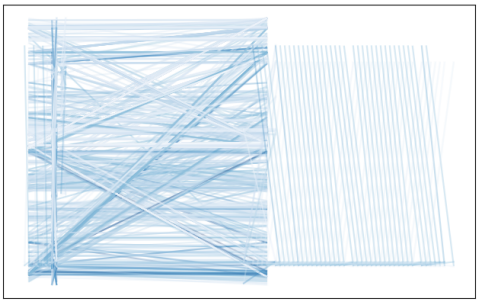

Figure 9: The dynamic graph from the visualization tool after using filters: weights between 0 and 100

It is clear that a lot of weights have been removed from the graph, which gives the user a totally different view of the data.

Now, the timestamps in the graph will be changed. The default for this is 0 to 300, this will be changed to 0 to 3. The colour will be changed from its default blue to the green colour scheme. This gives the following dynamic graph in Figure 10:

To explore the data in the visualization, it will come in handy to interact with the plots. That is why it is possible for the user to scour over the plot with the mouse and see the x- and y-coordinates. These are placed in the left below the plot. It is also possible for the user to cut out specific parts of the plot to see the dynamics more closely; this can of course be undone. The dynamic graph with a piece cut out of it can be seen in Figure 11.

It is also possible for the user to just visualize certain nodes from the data set. The user must make sure that the minimum and maximum weight and time do cover the time and weight of the wanted nodes, otherwise the plot will be empty. The nodes 61 and 32 will be selected and the dynamic graph with only these two nodes will be shown. These must be entered separated by a comma, the order of the nodes does not have any effect on the plot. Since these nodes are not covered by the parameter set, the parameters will be

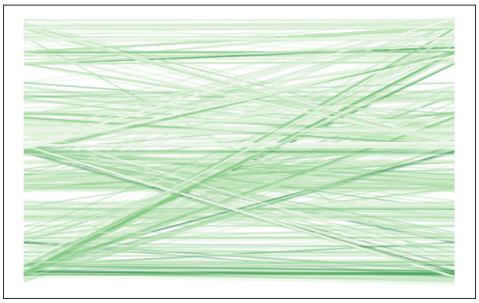

Figure 10: The dynamic graph from the visualization tool after using filters: weights between 0 and 100, timestamps between 0 and 3 and colour to green

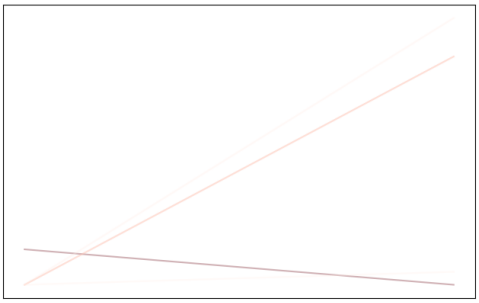

Figure 12: The dynamic graph from the visualization tool with default parameters after using node filtering, with nodes 32 and 61 and colour to reds

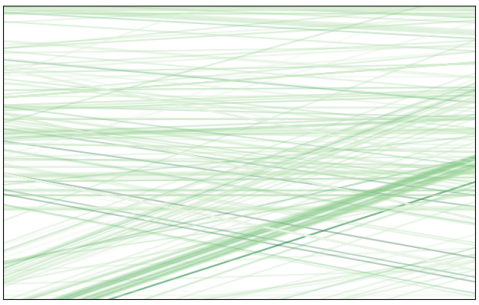

Figure 11: The dynamic graph from the visualization tool after using the cut-out function, with weights between 0 and 100, timestamps between 0 and 3 and colour to green

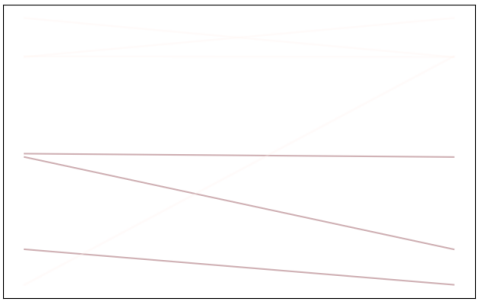

Figure 13: The dynamic graph from the visualization tool with Dijkstra's algorithm - start node 498, end node 841, colour to red

set back to their default, and use a different colour for the plot to show the difference.

The node filtering is turned off, which means that all nodes are included again so Dijkstra's algorithm can be applied on the data set. The beginning and end node that are filled in both must exist and need to have some kind of connection, otherwise the visualization tool will not update anything; the plot will simply remain the same. The start node is 498, the end node 841. This gives a new dynamic graph (Figure 13):

Running Dijkstra's algorithm also gives an output on the left side of the visualization tool. Here the weight of the shortest path from start to end node is given, which is 521588354 in the case that was just showed. Also, when running Dijkstra's algorithm, the output in Jupyter Notebook is updated and shows the nodes that the algorithm visits, which can be useful for some users.

As mentioned earlier, the GUI also has an adjacency matrix. This matrix changes along with the parameters, it only shows the data points that are also in the dynamic graph. However, an adjacency matrix on such a large data set often is incomprehensible. That is why a matrix reordering algorithm was included, reversed Cuthill-McKee, which makes the view of the adjacency matrix more useful for the user. This algorithm will be applied on the default parameters - without Dijkstra's algorithm - in the Reds colour map.

The user is offered the option to load a different data set in the GUI with just a simple click. Users are also given the possibility to save the plots that are made.

7 User Friendliness of the Tool

As described, one of the requirements of the tool was that it is easy to use. In this section, the tool will be evaluated on how user friendly it is.

7.1 Data Storage

Many tools require the user to have the data set in the same directory as the execution file, for example Jupyter Notebooks would not work unless the data set used is in the same folder as the notebook. This is quite annoying, since it requires the user to remember where exactly the files need to be stored to make it work. The user sometimes must store a particular data set in more than one directory, which is troublesome for the storage capacity of the PC, especially when dealing with data sets that require a lot of storage space.

The visualization tool created, however, can import data sets from all directories. The user can open all folders on their PC and it is also possible to upload data from the cloud, for example OneDrive or Google Drive. This gives the users more freedom in how they want to set up their directories and makes the tool very user friendly.

7.2 Exporting Graphs

Saving a created graph is often not possible, or a new line of code is required. Many users of visualization tools simply make a screenshot of the figure they want to use, having to crop out unnecessary parts of the figure. This results in pictures not being of equal size or in bad quality due to zooming in. A nice paper can easily be ruined by a bad figure.

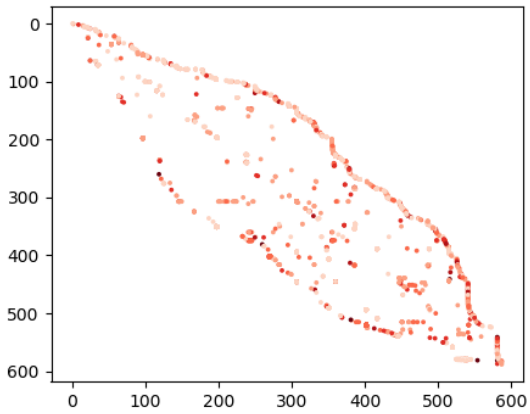

Figure 14: The adjacency matrix from the visualization tool with reversed Cuthill-McKee, default parameters, colour to red

The visualization tool gives, as shown earlier, users the possibility to save the graphs they created on their PC. This option considers the altered parameters and colours and the zoom-in function. This way, a users can always use the graphs they created with the tool for presentations or papers. With a simple click on the save icon, underneath the graph, the last opened directory on the PC is opened and the user can save the picture. Of course, the user can then open another directory and name the picture, making sure that the picture can be found very quickly. The saved figure is of high quality, still showing all details and dynamics of the data. With this option, the user-friendliness of the tool is increased.

7.3 Performance Test

A good visualization tool can process big amounts of data within seconds, without losing its accuracy. This is a trade-off that was considered whilst making the tool. To test how well the tool performs, random generated data was uploaded in the tool. These data sets grow exponentially in size, and the time it took the tool to open with this data set was measured and put into the table below (Table 1).

Size of data set ENTITIES	**Running time** SECONDS
10	< 1
100	< 1
1,000	1
10,000	71
100,000	1274
1,000,000	CANNOT BE RENDERED

Table 1: Table showing how long the tool takes to process data sets of a particular size

It is clear that the tool performs well on smaller data sets, but when a data set contains over 10,000 entities, the tool becomes very slow. The tool clearly cannot handle big data sets, which is unfortunate at this time considering the huge amounts of data available. A user cannot visualize a big data set with this tool, which makes the tool less user friendly.

7.4 Launching the Tool

Want to use a tool: easy. Just type in the name of the tool on the computer and it will be found, most of the times even after typing in the first letters. Launching a tool like this will cost a user not even half a minute.

However, this is not the case for this visualization tool. Due to the modules used in the code and the lack of knowledge on the topic, it was not possible to make the tool really web-based. The tool can only be launched by opening the Jupyter Notebook and then running the code. This takes often more than a few minutes, since the correct notebook must be opened in the Jupyter launcher. The code needs to run every time a user wants to work with the tool. However, the code is over 500 lines long and takes quite a while to run. Another implementation could have prevented this. This wastes valuable time and harms the user-friendliness of the tool.

8 Workflow

The workflow around this project can be divided into two parts. Firstly, the process of solving this project step-by-step; how to start with nothing and how to work towards finishing parts of the bigger picture. Secondly, the visualization pipeline of the GUI which describes the main principle of the GUI.

8.1 Process

The final GUI was not built in one day. It was made over an 8-week long quartile, in which process has been made almost every week. At the start of the quartile, everything was moving at a slow pace which helped the team to get a clear concept of what is wanted. But after two weeks of making introduction like exercises, the first GUI assignment was given in the third week. The entire group gathered and started drawing ideas on the whiteboard on the visuals of the GUI. This had to be implemented in code in the same week which was difficult, but it did not have to be functioning at its finest. Therefore, some GUI code was written with the main purpose of getting to know the program and implement basic functions like file loading, implementing an adjacency matrix, a dynamic graph visualization, and Dijkstra's algorithm.

In this week, code for all these functions was made but the GUI was not very appealing nor functional. Then the following week, filters for different dimensions like time and edge dimensions were applied. There was also an added possibility to change the colour of the graph. However, the node-link diagram which had to be made this week was not quite finished before the deadline nor did it provide the details wanted. This was mainly due to the large data set.

Then a matrix reordering algorithm was implemented together with the wanted interaction techniques. The hardest part appeared at this point, the made code for everything had to be implemented into the GUI to make it one project. While implementing the code into the GUI, some code did not work properly or something was missing from the ideal picture of the GUI. Therefore, to meet all deadlines and to implement the code perfectly, the team split up into two teams; one team would only focus on implementing code and improving the GUI, the other team would work on the remaining group assignments, being loading different data sets and performing a performance test.

8.2 Visualization Pipeline

To create the visualization tool, the following visualization pipeline is used. This visualization pipeline, as shown in Figure 16, gives insight in how a data set gets transformed into visualizations in the Graphical User Interface.

9 Discussion and Limitations

During this 8-week long quartile, the group worked very hard on achieving all goals that were set in the course. A lot of effort was made to get the GUI up and running, making it look nice and having all interactions implemented. Unfortunately, implementing all loose parts of code that worked on their own, did not always optimally work in the code with over 500 lines that made the GUI work for the user. It costed the group a lot of time to fix errors that were not

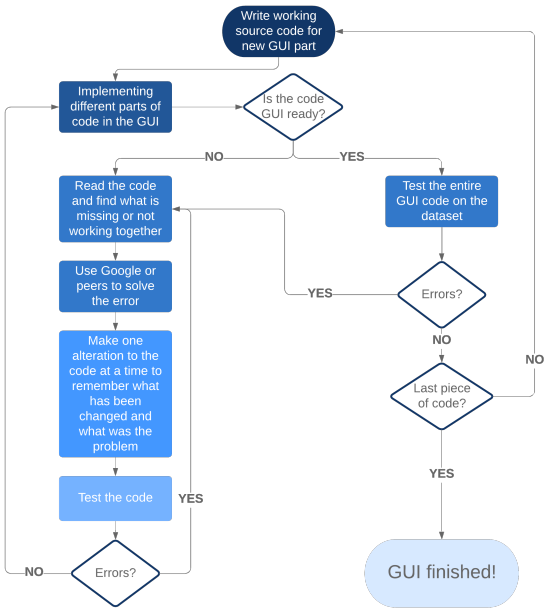

Figure 15: Workflow chart of the project [15]

expected but had to be solved to move on to other parts. This made it happen that the final GUI is not as complete as the group wanted it to be.

To start the visualization tool, the user needs to run the source code in the Jupyter Notebook. This is unfortunate, not only because the users need to have the source code ready on their PC, also because this makes the tool not really web based in the end, one of the requirements for the tool. The module used for the GUI, tkinter, does not make it possible for the GUI to be web-based, which is why this limitation occurred. This is something that the group could have changed during the process, by switching to the module dash [17] for example. However, since the course lasted only eight weeks, the group members decided that it was more important to implement interactions and algorithms than to switch to dash, which would have costed a lot of time.

The source code for the visualization tool also has a big running time. When running the code under default parameters, so not the entire data set is taken into the GUI, it often takes over 30 seconds for the tool to build. When the entire data set is used, it takes over ten minutes. This makes the tool not usable in daily life, especially when working with big data. We, however, have no clue why the code takes this long, which is why it has not been changed throughout the course.

Most of the graphs in the visualization tool were made with the networkx module. However, after updating Anaconda Navigator, the networkx function that plotted all these graphs did not work anymore, due to removal from its owners [18]. This happened to three out of five laptops that were used in the project, making it very hard for the members to work on the code because of the constant errors that could not be solved. The GUI was then also finalized on just one laptop. Considering the newest version of Anaconda Navigator does not allow the user to make networkx graphs, and the code needs to be provided to make the tool work, it is hardly possible that the tool can be used by someone who did not have Anaconda Navigator before its update. A user will then always have errors that cannot be solved. Since the deletion of the function is not the fault of any of the group members, there was very little that could be done by the group members, but it really limits the usability of the visualization tool.

10 Conclusion and Future Work

This report described how the GUI can transform a data set into visualizations to gain information. The layout of the GUI, with its interactions and adaptable parameters, was explained and shown in much detail. The choices that were made were criticized and the pros and cons of these choices were discussed.

10.1 Conclusion

During this course, all group members put in a lot of effort to finish this course with a good-looking GUI. None of the group members has ever made a GUI before, which made the project challenging and new. It also made it hard since there were no bases of knowledge about such a project. Google was used to do research on the best modules and visualizations for the tool. A lot of time was spent on finding out how to start making a plain GUI, and then the visualizations had to be implemented [2].

10.2 Future Work

The end of the course has been reached, however, this does not mean that the visualization tool is 100 % finished. On the contrary, there are still many things that can be changed for the better, and even more new things that can be implemented in the tool.

10.2.1 Improvement of the Graphical User Interface

One point of improvement is concerning uploading different data sets. The visualization tool was made using the data set provided within the course. This data set consists of four variables, time, start, target, and weight with all numerical values assigned to these variables. The code for the tool is also based on these four variable names, which makes the tool unable to work with a data set that does not have these four variable names. To upload different data sets, one must, before uploading the data set in the GUI, change the variable names of the data set to the following names: time, start, target, and weight. In the future a function must be written which automatically changes the variable names of different data sets to names with which the GUI can function properly, or the code must be altered such that the tool accepts all data sets.

Another point of improvement is the node-link diagram. The node-link diagram now visualizes all nodes in the data set which leads to an enormous amount of data being loaded into a small rectangle reserved for the node-link diagram. This causes a huge number of crossings between the nodes and therefore, it is hard to easily extract information from this visualization. To improve this, the node-link diagram can be transformed into a hierarchical, radial, or force-directed node-link diagram. In this way, all the data can be visualized without losing the ability of extracting information from the visualization. If this could be implemented, it could be improved by adding extra visual variables, like changing the line colour depending on the weights of the vertices and add direction in the vertices by using arrowheads. Textual information can also be implemented using mouse-click interaction. In this way, it does not always disturb the visualization but when one hovers with the mouse over something that has a label, the label pops up, so the labels can be seen.

An attempt at making a graph like this was done in the process, but due to lack of time this was not implemented in the tool. An easier method to improve the node-link diagram is to make it parameter-dependent, like the other two graphs in the tool. This can already make the graph clearer and comprehensible, and is not hard to implement.

10.2.2 Extensions of the Graphical User Interface

In the future, a broad range of elements can be added into the GUI. Many new algorithms can be implemented, for example shortest path

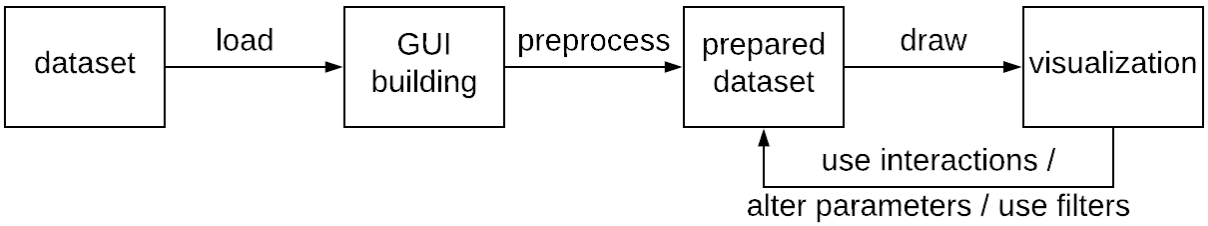

Figure 16: Visualization pipeline [16]

algorithms (FloydWarshall's Algorithm [19] and Bellman Ford's Algorithm [20]) or matrix reordering algorithms (a greedy algorithm and a clustering algorithm [21]). With Floyd-Warshall's Algorithm, the weights of all pairs of vertices can be calculated. With Bellman Ford's Algorithm, the shortest path between vertices can be calculated for data sets with negative weights between the vertices. The extra options for matrix reordering give new insights in the data. All these extra algorithms can be useful for users, considering they all want to discover different things in their data. It also makes it possible for users to use data sets with a different setup than the data set that was used. These options give the visualization tool a lot more opportunities.

Another extension of the GUI should be adding another visualization type, being an alluvial diagram [22]. Alluvial diagrams show how various nodes flow together or apart across time. In these diagrams the width of stream shows the weights of the nodes and patterns of nodes can be visualized. This diagram would be another option for the dynamic graph visualization and it would replace this visualization if the added Alluvial diagram-checkbox has been checked and the filters were to be updated. The addition of an Alluvial diagram would be useful since it can visualize flow like behaviour in data and it can visualize a lot of data without the visualizations becoming incomprehensible. It gives a complete look on the data set while at the same time allowing users to interact with the diagram and see the relation between certain nodes [23].

Acknowledgements

First, we would like to thank all our group members. Because of our combined work and all the meetings, we completed everything that we wanted to do. Furthermore, we would like to thank dr. M. Burch for all the work he put into teaching us, responding quickly to our e-mails, and to give proper feedback on our assignments.

References

[1] Michael Burch. Isoline-Enhanced Dynamic Graph Visualization. Proceedings of 20th International Conference on Information Visualisation, IV, 1-8, 2016

[2] Christoph Müller, Guido Reina, Michael Burch, Daniel Weiskopf. Large-Scale Visualization Projects for Teaching Software Engineering. IEEE Computer Graphics and Applications 32(4), 14-19, 2012

[3] Michael Burch, Benjamin Schmidt, Daniel Weiskopf. A Matrix-Based Visualization for Exploring Dynamic Compound Digraphs. Proceedings of 17th International Conference on Information Visualisation, IV, 66-73, 2013

[4] Michael Burch, Marcel Hlawatsch, Daniel Weiskopf. Visualizing a Sequence of a Thousand Graphs (or Even More). Computer Graphics Forum 36(3), 261-271, 2017

[5] Michael Burch, Gerald Baulig, Tobias Boley, Arjana Mehmeti, Dina Kurbanismailova, Marc Roswag, Oliver Streicher, Steffen Wittig, Uwe Kloos. IMDb Explorer: Visual Exploration of a Movie Database. Proceedings of the 11th International Symposium on Visual Information Communication and Interaction, VINCI, 88-91, 2018

[6] Fabian Beck, Michael Burch and Stephan Diehl. Matching Application Requirements with Dynamic Graph Visualization Profiles. Proceedings of 17th International Conference on Information Visualisation, IV, 11-18, (2013)

[7] Dataversity. (2016, January 12). *The Importance of Big Data and Data Visualization.*. Retrieved from https://www.dataversity.net/the-importance-of-big-data-and-data-visualization/

[8] Computer Hope. (2018, November 13). *What is a Graphical User Interface (GUI)?* Retrieved from https://www.computerhope.com/jargon/g/gui.htm

[9] IMDb. (n.d.). *IMDb Datasets.* Retrieved from https://www.imdb.com/interfaces/

[10] Bezerra Beniz, D. & Espndola, A. (2016). *Using Tkinter of Python to Create Graphical User Interface (GUI) for Scripts in LNLS.* ISBN 978-3-95450-189-2

[11] Diehl, J. (2004, January 19). *Preprocessing and Visualization.* Retrieved from http://www.keysers.net/daniel/files/DiehlSlidesReport.pdf

[12] Grace - Martin, K. (2018, December 15). *When List-wise Deletion works for Missing Data.* Retrieved from https://www.theanalysisfactor.com/when-listwise-deletion-works/

[13] Sniedovich, M. (2006). *Dijkstra's algorithm revisited: the dynamic programming connexion.* Control and Cybernetics 35.3 (2006): 599-620.

[14] Liu, W., & Sherman, A. H. (1976). *Comparative Analysis of the Cuthill McKee and the Reverse Cuthill McKee Ordering Algorithms for Sparse Matrices.* SIAM Journal on Numerical Analysis, 13(2), 198-213. doi:10.1137/0713020

[15] Lucid Software Inc. (2018, August 30). *Workflow Diagrams Landing Page.* Retrieved from https://www.lucidchart.com/

[16] Boezer, M., Cornelissen, J., Hofland, R., Klaassen, N., Verhoeven, J., Wesselink B. *Generating Hierarchical Structures: Web Visualizations* Retrieved from https://canvas.tue.nl/courses/8958/files

[17] Plotly. (2017, June 21). *Introducing Dash.* Retrieved from https://medium.com/@plotlygraphs/introducing-dash-5ecf7191b503

[18] NetworkX. (2018, September 19). *Drawing.* Retrieved from https://networkx.github.io/documentation/stable/reference/drawing.html

[19] Hougardy, S. (2010). *The Floyd Warshall algorithm on graphs with negative cycles.* Information Processing Letters, 110(8-9), 279-281. doi:10.1016/j.ipl.2010.02.001

[20] Bahadur Singh, J., Tripathi, R. C. (2018). *Investigation of Bellman Ford Algorithm, Dijkstra's Algorithm for suitability of SP.* International Journal of Engineering Development and Research (IJEDR), 6(1), 755-758. http://www.ijedr.org/papers/IJEDR1801130.pdf

[21] Behrisch, M., Bach, B., Riche, N. H., Schreck, T., & Fekete, J. (2016). *Matrix Reordering Methods for Table and Network Visualization.* Computer Graphics Forum, 35(3), 693-716. doi:10.1111/cgf.12935

[22] Green, E. (2018, August 01). *Alluvial Diagrams Towards Data Science.* Retrieved from https://towardsdatascience.com/alluvial-diagrams-783bbbbe0195

[23] Yeung, A.W.K. (2018, May 13) *Data visualization by alluvial diagrams for bibliometric reports, systematic reviews and meta-analyses.* Current Science, 115(10), 1942-1946. doi: 10.18520/cs/v115/i10/1938-1942