Building an Interactive Visualization Tool for Dynamic Graph Datasets

Rens v. Eggelen[*], Tom van Knippenberg[†], Wessel Krenn[‡], Tom v. Meer[§], Marko Petkovic[¶], Bart Slenders[‖], Michael Burch[**]

Eindhoven University of Technology

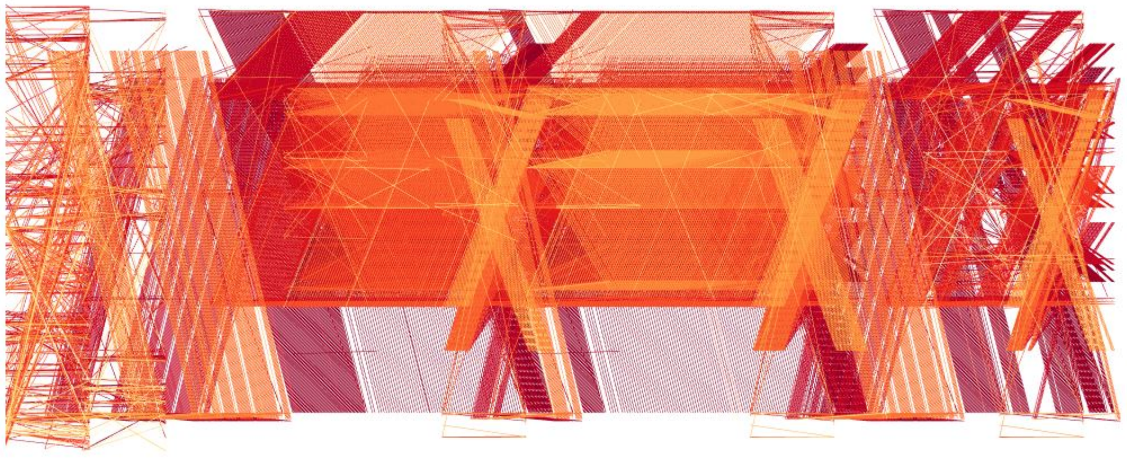


Figure 1: The profile semantic trafo final dataset visualized with an interleaved dynamic network graph visualization

Abstract

This paper describes an interactive web-based tool for visualizing dynamic graph data. The tool supports experts to upload, explore, and compare dynamic graph data using a multitude of different dynamic graph visualizations as well as perform algorithmic graph problems and filter data. The interactive tool makes use of the Python programming language and Dash, a Python framework for building analytic web applications, and runs in a web browser. We show the visual output of a dataset in a way that is intuitive for the users to navigate and the tool allows them to visualize their data easily. In this paper, a case study will be provided in which we show an example of the application, the limitations of the tool will be discussed and finally, there will be a conclusion with recommendations for future research.

Index Terms: Human-centered computing—Visualization—Visualization techniques—; Human-centered computing—Visualization—Intuitive graphical user interfaces—; Human-centered computing—Visualization—Visualization design and evaluation methods

[*]e-mail: R.S.v.Eggelen@student.tue.nl
[†]e-mail:T.A.H.v.Knippenberg@student.tue.nl
[‡]e-mail: W.T.Krenn@student.tue.nl
[§]e-mail: T.A.J.P.v.Meer@student.tue.nl
[¶]e-mail: M.Petkovic.1@student.tue.nl
[‖]e-mail: B.F.Slenders@student.tue.nl
[**]e-mail: m.burch@tue.nl

1 Introduction

Dynamic graph data is everywhere [3]. Algorithms that call functions within itself [9, 15], social circles (in social media) [4], email networks within organizations, workflows [22], or flight networks [8, 12], build an endless list of applications that can be expressed as or simplified to dynamic graph data. Data sets of this type can contain millions of elements that are connected to each other [2].

The world is constantly evolving, generating more data every second, adding to the size of databases. With current storage technology, it is cheaper to save data, than to figure out which data to delete. Companies exist with the sole purpose of obtaining useful information from these large amounts of data. They do this to find solutions to complex problems, to gain an understanding of the world and perhaps to make it a better place, or just to make a profit.

To do this, said companies can no longer just look at the textual data as it is really hard to see patterns. Therefore, there is a need to visualize these datasets to accommodate the understanding of the data, as well as the rapid identification of patterns within said data.

Visual representations allow for the filtering and exploration of the datasets [10], as well as the running of algorithmic graph problems on it, which is a requirement for the data to be useful. The graphical representation of network data is a relatively new scientific field that is said to originate in 1735. Its roots can be traced back to Leonard Euler who worked on the problem of the Seven Bridges of Königsberg and created a visual approach in terms of a mathematical network that would work best to solve the problem. The fact that this is a new research field means that there are still new techniques being researched to this date, making it a very active field.

This paper aims to take knowledge from different fields to create and introduce an interactive, web-based dynamic graph data visualization tool [1] that can help users process their textual dynamic graph data into a graphical representation, allowing them to learn from the data.

2 Related Work

The ability to design a user interface that attracts the user's attention and enhances the user experience is a challenge that the GUI designers face. The aesthetics of the user interface is the predominant factor in getting the users' attention. Careful implementation of aesthetic concepts can aid learn-ability and acceptability from the users side. Studies showed that very high correlations were found between users' perceptions of interface aesthetics and usability because if a page looks clean to a users' eye, they will find it easier to work with or on the page. For example, radial versions of visualizations [11, 13, 32] are considered more aesthetically appealing, but on the other hand, they may create perceptual issues to the data analyst.

Eye tracking is a powerful concept to understand where and when people pay visual attention [6, 7], however, the power strongly depends on the application field, the stimuli, and the tasks at hand [25, 27–30].

To make dynamic graph data explorable we need an intuitive and understandable user interface based on visual analytics concepts [24] that follows the visual information seeking mantra: overview first, zoom and filter, then details-on-demand [31]. For this reason we followed a left-to-right reading direction by first letting the user change parameters that are visible on the left hand side while the impact of the changes is shown in the center view, similar to the approach by Bruder et al. [8] or Burch et al. [16]. The right hand side view is used for details-on-demand.

To benefit from different views on the dynamic graph data we have designed a visualization tool combining dynamic graph visualizations as an overview about the dynamics of the data, but also additional views like adjacency matrix visualizations [32] as well as node-link diagrams in different layouts [19, 23]. The views are linked interactively.

Moreover, we support matrix reorderings [5] as well as finding the shortest path in graph data while the evolution of the shortest paths is also highlighted in the dynamics of the graph [20]. Filter options can be applied based on typical graph data dimensions like vertices, edges, time steps, or weights, but even on the detected shortest paths, only showing the vertices and edges in those paths.

3 Data Model and Preprocessing

To be able to use particular data files in this visualization tool, the file has to be in a particular format. This format is a plain text file with on each row a time, a start node, a target node, and a weight value, for example: '1 842 980 604799'. This means that on time stamp number 1 node 842 has a connection with node 980 with a weight of 604799.

Furthermore, there cannot be any white spaces after the link weight or white lines between rows when the data file is uploaded into the tool. The tool checks whether this format is correctly applied and splits it on every line afterwards. This is all to ensure that the uploaded data works with the implemented algorithms, which use this .txt file.

The tool accepts two other formats of data: a gzip csv file or a gzip dat_ file. The tool converts these files into a .txt file with the same format as is described above. However, it is adjusted to the specific files that were found on the internet. So, any other file with the same file extension may lead to weird results. This would be improved by asking the user what the order of the numbers is, so either the time, start, target, or weight. This is also described in the conclusion and future work. The uploaded file with the extension is then deleted from the tool and what is left is the filename in txt format. The file is then ready for use in the tool.

First, when the text file was opened, it was converted into a three-dimensional matrix, where the start node and target node are two of the dimensions, the time is the third and the weight value is just stored as a number on that spot in the matrix.

After bigger datasets came in, it became apparent that this conversion took too long for the tool to handle these datasets. Therefore, the conversion into a three-dimensional matrix was changed into a three-dimensional sparse matrix where the zeros are ignored. Thus, in a bigger dataset with many zeros this is computationally more efficient.

4 Visualization Tool

In the following we will describe the visualization tool with the graphical user interface and the functionalities.

4.1 Graphical User Interface (GUI)

To meet the general principle of a good GUI design as stated above the standard provided template was chosen because it would fit the chosen frameworks and packages optimally. It also meets the principles simplicity, consistency, comprehensibility, clarity, and aesthetically pleasing (see Figure 3). The layout is focused on being intuitive for the user in that the experience is designed around a logical chronological order of operations. The template design was chosen because of the minimalistic style keeping the visual clutter to a bare minimum and help generated visualizations stand out, allowing the users to direct their focus onto the most important bits of the page.

In the Western society, it is customary to read from left to right, from top to bottom. The GUI is read the same way. The user starts in the upper left hand corner selecting a dataset. After switching to the interactive dashboard, the visualization technique can be chosen via a drop-down menu which is easily readable and instantly performs its task when clicking on it, giving the user a sense of control. The user can make a choice between different visualizations. The selected visualization is shown as a graphical representation that can be found in the center of the screen, which is the central point of focus as well. The visualization appears in the biggest available space on the screen drawing immediate attention of the user. On the right hand side of this graphical representation, some space is reserved for a future implementation of extra information like the values of a node or link the user is hovering over, or textual answers to graph problems that have been applied to the dataset. Displaying this in the same page of the visualization itself helps the user maintaining overview and keeps the interface clear in visual appearance by reducing visual clutter and keeping things understandable.

In the left hand bottom corner, there is a navigator view that displays the selected dataset as an adjacency matrix, allowing the users to view their data in a different way while maintaining overview. This meets the clarity principle. An example of this adjacency matrix with the profile trafo semantic final dataset selected can be found below in Figure 2.

Next to this navigator view and below the visualization, the user is able to set parameters via a selection slider. Note that these sliders are not yet visible when no dataset is selected meeting one way to provide simplicity by using progressive disclosure, hiding things until they are needed. These selection sliders are a tool for the user of the interactive visualization tool to select the node-range for the selected graph above. Below this slider, two other sliders can be found. One for selecting the log weight range and the other for selecting the time range or time stamp depending on the selected visualization. These sliders help the user intuitively set boundaries based on the visual data above. Before the new settings of these sliders are applied, the user must first click on the execute changes button located on the left of the visualization. When a user executes the changes that he or she has made, a loading circle will appear in front of the main visualization indicating that the system is working on refreshing the visualization with the selected parameters specified by the user.

Lastly, a console log will be added in the bottom right hand corner of the screen, allowing the users to log their progress by showing them what they have already ran, what they are running at the moment and giving them potential warnings when the user sets filters that leave no data or when the system is about to crash due to a too big dataset. It should be noted that this feature has not been implemented yet.

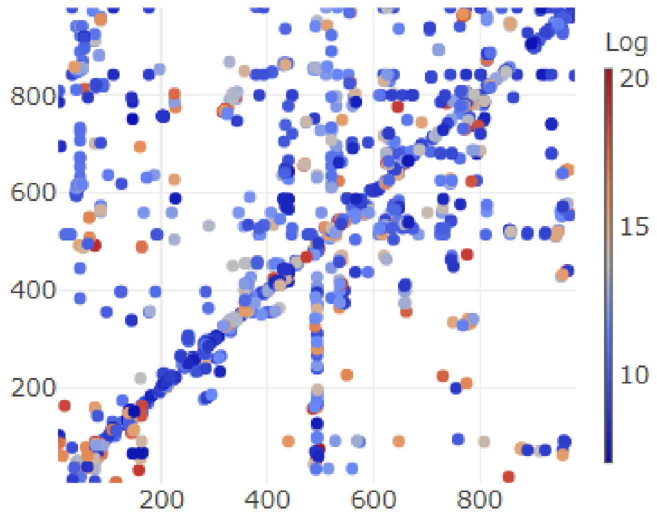


Figure 2: Adjacency matrix as navigator in the visualization tool

4.2 Components

The tool consists of two main components. The first being the upload page, where the users of the visualization tool can choose their dataset either from their computer or from the overview of previously uploaded datasets. Once a dataset has been chosen and uploaded, the user can view a summary of the dataset he or she uploaded below in the drop-down menu. This shows the head of the data with the columns: time, start, end, and weight.

The second component is the interactive dashboard, where the user can choose three different visualizations to visualize the data. This page in turn consists of 6 components, starting top left there is an options menu where the users can select one of the uploaded datasets, the type of plot they want plotted, a color scale, an option to execute Dijkstra's shortest path algorithm from a starting to an end node, and the option to re-order the adjacency matrix down below. Then top middle is reserved for the main visualization view, this is where the selected plot from the previous compartment will be shown. The top right is a space reserved for a not yet implemented details-on-demand view where the users can see information about a node and its friends when they hover over a node in the main visualization view or the navigator view, of which the last one is located on the bottom left hand side. Right next to this navigator is a compartment for a slider menu where three different sliders will appear once an uploaded dataset and a plot have been selected in the top left part of the screen. Here the user can filter out nodes, timestamps or weights of edges to their liking. And last but not least in the box on the bottom right, there is a not yet implemented console log to see how long it took to generate a selected plot for example.

4.3 Visualization Techniques

The users can choose three distinct visualization techniques to visually represent their data. There are three choices of which the first is a radial plot. In this plot the data is visualized as a circle, in which the nodes form the outside layer of the circle and the edges create a spiderweb-like visualization on the inside of the circle. This is a rather intuitive way of visualizing the data, though it does suffer from visual clutter, especially with datasets which contain a lot of edges connecting nodes. An example of this radial plot can be found in Figure 4.

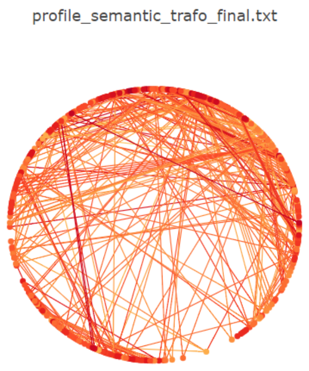


Figure 4: A radial plot on the profile semantic trafo final dataset with a different color scale [yellow-orange-red]

A Fruchterman-Reingold plot, which is a fast force-directed layout algorithm, in which the sum of the force vectors determines the position of the nodes. This plot is very suitable for executing and reviewing a shortest path between nodes, also called Dijkstra's shortest path algorithm. An example of this plot with a shortest path can bee seen in Figure 6. Note that the path is highlighted and the rest is scaled back in color intensity.

Finally, an interleaved dynamic network visualization (Figure 5) is implemented that can be used to display, navigate, explore, and analyze dynamic graph data with a thousand or even more time steps. It is static, aiding the user to represent it mentally. This plot is very good for picking out patterns in the data for further examination. The running time of this plot is quite long and the complexity is quite high, but it was managed to scale down the running time, generating plus plotting, from 12 minutes to 20 seconds on the profile semantic trafo final dataset.

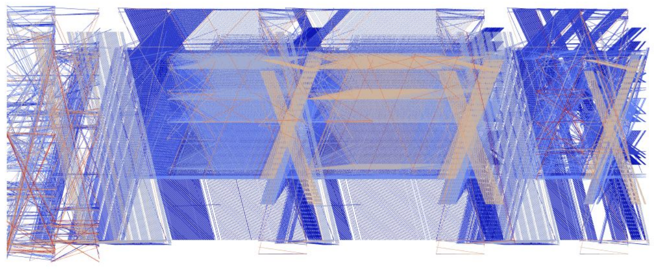


Figure 5: Three different color codings can be used in the tool; here the default blue/red

Also, no matter the user's choice of visualization technique, a navigator view in the form of an adjacency matrix is available for easy navigation in the graph. This visualization scales with the main visualization when filters are being applied and it is also possible to re-order this matrix with a simple click of a button right above it. This is shown in Figure 7. The algorithm that is used to perform this reordering can be found in Section 4.4. The resulting reordered adjacency matrix will always be in the form of a leaf.

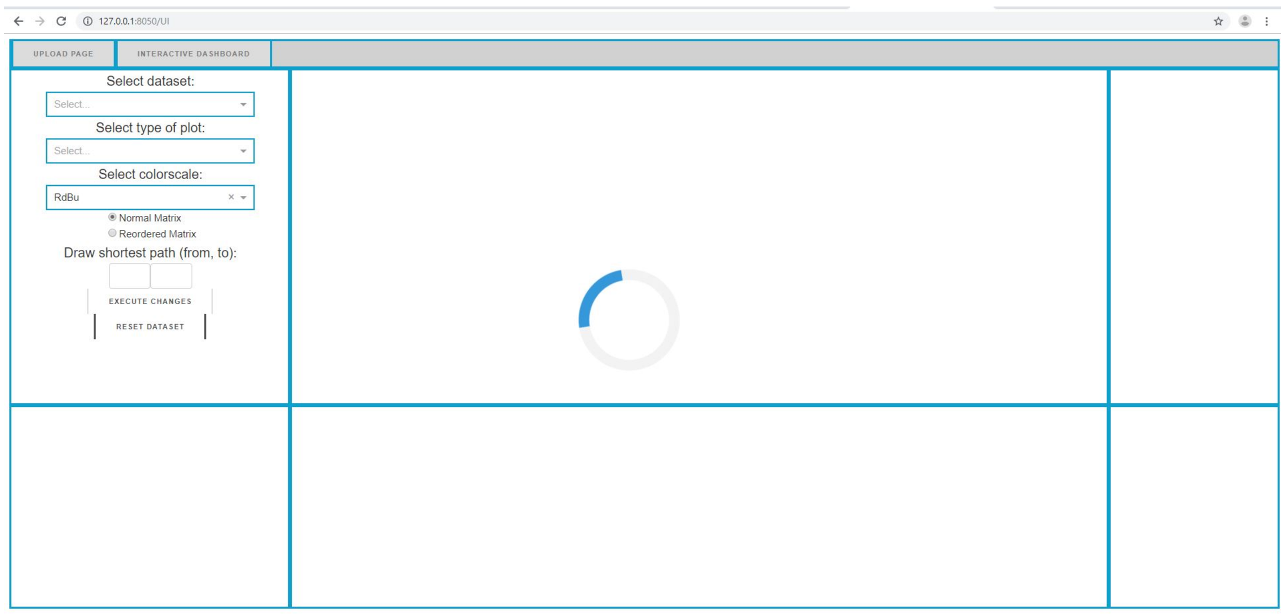


Figure 3: The interactive visualization tool dashboard without a dataset loaded into it where you can get an overview of how the dashboard looks without the options to filter data

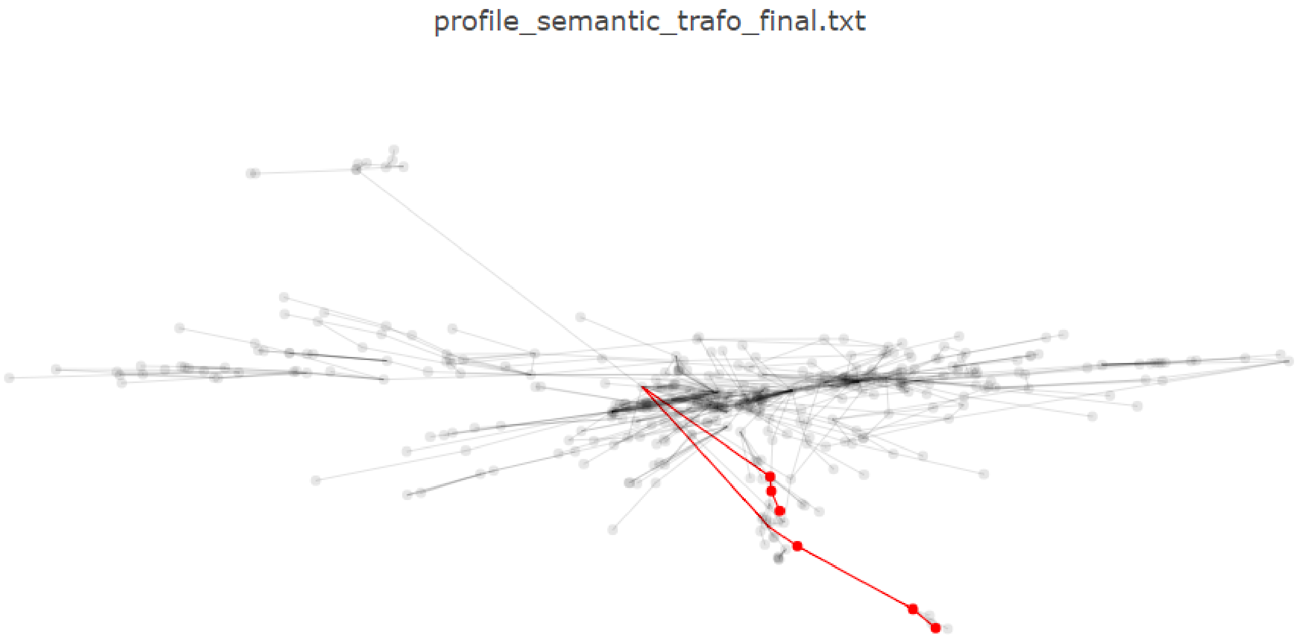


Figure 6: A Fruchterman-Reingold plot on the profile semantic trafo final dataset with Dijkstra’s shortest path algorithm applied to it from node 715 to 107 (in red)

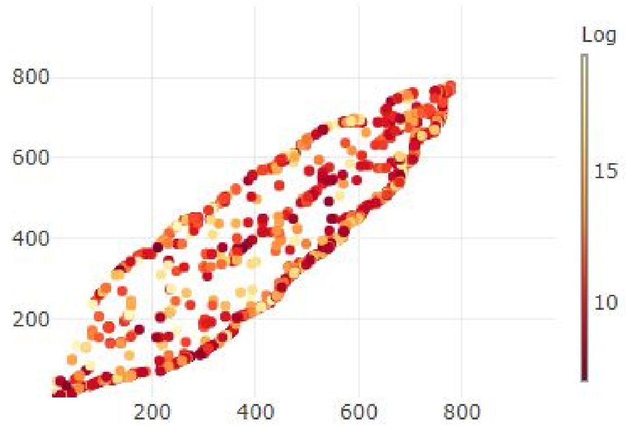


Figure 7: Reordered adjacency matrix with use of Reverse Cuthill McKee algorithm with the color scale [yellow-orange-red]

4.4 Algorithms

First of all, a standard Dijkstra's algorithm is used to calculate the shortest paths in the datasets, which can do both simple short paths spanning two nodes as well as more complex paths. To make the shortest path stand out and increase visibility, the shortest path gets a different color from the other edges. In dense graphs, this can still be difficult to see. To make the shortest path stand out even more, the opacity of the other edges is decreased. This makes sure the red colored shortest path is clearly visible no matter how dense the graph is.

The plotting algorithms work in a procedural, meaning line by line, manner by reading data directly from the text file. The algorithm will go over every entity while plotting it. This is an improvement over the previous approach, where a three-dimensional matrix was used to store the data. The three-dimensional matrix was 800 by 800 in size per point in time. With 1,000 time points, this quickly adds up. Not only is this inefficient in using storage space as many points may be empty or zero, it also has a slower running time. The new procedural approach is beneficial in terms of complexity, use of storage, as well as running time.

The matrix reordering algorithm does not use this procedural method, but uses a sparse matrix to store the data instead. This is a similar approach to the three-dimensional matrix. In this case, the many empty or equal-to-zero points are a benefit, which decreases the running time of the matrix reordering algorithm. The reordering function makes use of the Reverse Cuthill McKee reordering algorithm, as described by Behrisch et al. [5], is used to perform permutations on the adjacency matrix. The outcome is an adjacency matrix that has a certain bandwidth; it looks like a leaf. An example of a reordered adjacency matrix can be found in Figure 7.

The first graph is a radial plot, as can be seen in Figure 4, which is created by a self written algorithm that places all the nodes in the dataset on the outer line of a perfect circle. Starting off with the right half of the circle and then plotting nodes with a higher number in a counterclockwise manner. This takes advantage of the properties and layout of most datasets, decreasing running time drastically. The color of the plotted nodes are blue/red by default but can be changed by changing the color scale to greys or yellow/red tones as can be seen in the image on the front page of this paper. The blue/red color scale can be found in Figure 5 and the greys color scale can be found in Figure 8 where both the main visualization and the adjacency matrix change color after applying the color scale of choice of the user.

Secondly the Fruchterman-Reingold plot can be seen in Figure 6. The Fruchterman-Reingold algorithm calculates the sum of force vectors, which determine the directional movement of a node. The step width is a constant and determines the distance nodes move in a single step. The nodes stop moving when the equilibrium state is reached, which is when the energy is minimized. This is a random procedure, therefore performing this visualization technique twice may lead to two different outcomes. This is not ideal for the mental representation of the user, that is why the radial plot was added to the options of visualization techniques.

Finally, the interleaved dynamic network visualization works by a specific pre-processing method in which vertex ordering, clustering, and then applying a novel interleaving approach, based on scalable visualization approaches make it a suitable choice to work with. Time is plotted on the x-axis and the nodes are placed from low to high. With this visualization technique it is easy to detect patterns in the data, for example where the most interactions take place over time.

4.5 Interactions

All seven categories of interaction (select, explore, reconfigure, encode, abstract/elaborate, filter, and connect) have been implemented in this visualization tool [33].

Select interaction techniques provide users with the ability to mark (a) data item(s) of interest to keep track of it. In this visualization tool, this can be achieved by uploading a data set and selecting it. The title above the plot marks it as the chosen set. There is also the option of hovering over data points to select them.

Explore interaction techniques enable users to examine a different subset of data cases, typically by examining a subset of the data to gain understanding and insight, before moving on to other data. Plot.ly offers a lasso tool to select a subset in the plots, allowing users to explore.

Reconfigure interaction techniques provide users with a different perspective onto the dataset, by changing the spatial arrangement of representations. In the case of this visualization tool, users can choose between different visualization techniques to gain new insights. Also, the option to reorder the adjacency matrix is an example of a reconfigure interaction technique as it shows a different version of the adjacency matrix that can lead to new insight into the data.

Encoding means to enable users to alter the fundamental visual representation of the data, like the visual appearance of the data elements. This is achieved by allowing the users to select different color codings for the navigator view. There are three options: Greys, Yellow or Red, and Red Blue.

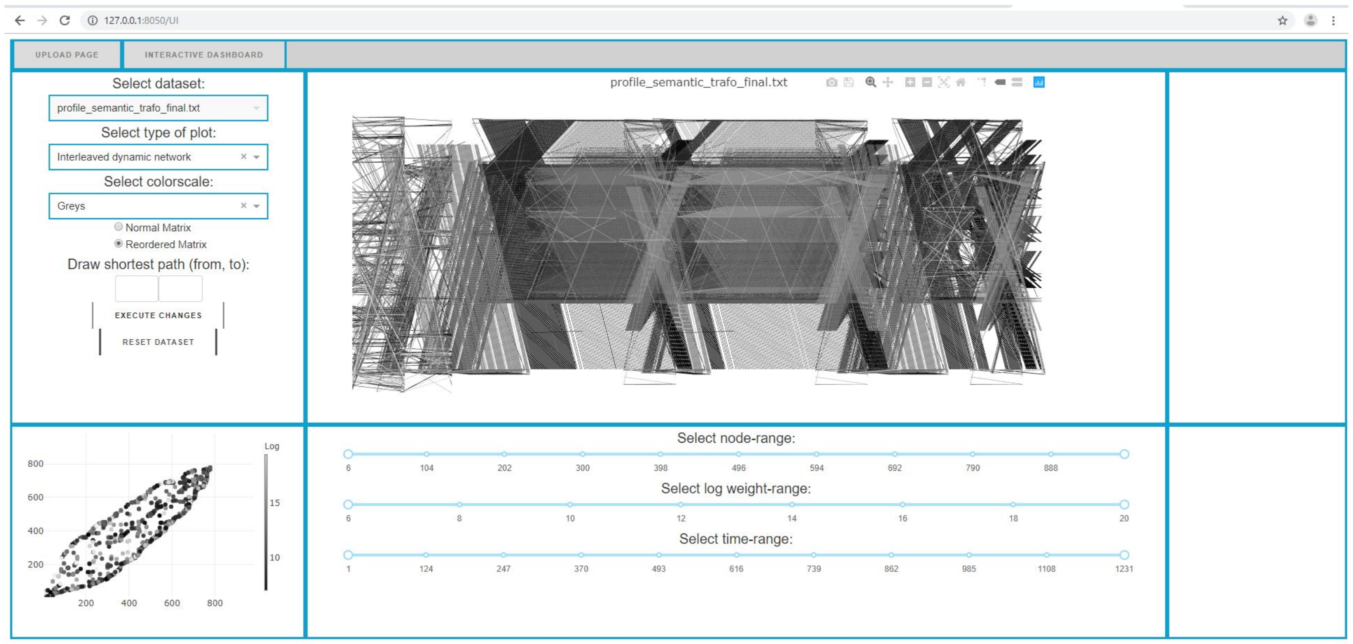
Abstract/elaborate is the providing of the users with the ability to adjust the level of abstraction of a data representation. In this case, this is done by allowing users zooming options as provided by Plot.ly.

Filtering is showing the user something conditionally, which users are allowed to do by using the filter sliders below the main visualization. The three sliders for time, nodes and weights provide this.

Finally, connecting is the process of highlighting associations and relationships between data items that are already represented. This tool achieves that by the option of drawing a shortest path in between two data items, if there is one. This feature has been discussed before.

5 Application Example

The main idea of having a visualization tool is to easily recognize patterns, in this case, in dynamic data sets as already mentioned in the introduction and visualization tool part of this report. Users have a significant amount of textual data that they cannot begin to analyze without a computer. Even with a computer, the size of the data is overwhelming and finding points of interest may be


Select dataset: profile_semantic_trafo_final.txt

Select type of plot: Interleaved dynamic network

Select colorscale: Greys

Normal Matrix [✓ Reordered Matrix]

Draw shortest path (from, to):

Figure 8: The graphical user interface with an interleaved plot in the main part and all the features

impossible. Recognizing patterns, links, correlation, and causation may be beyond impossible. This is where visualizations come in.

A good example would be email traffic in a customer support center. If a user, a manager for example, is interested in how many emails each of his/her employees send, whom they send the emails to, at what times most emails are sent or received, and how long it takes employees to reply to emails during each working day, this manager could use this visualization tool.

The manager could then possibly find patterns in the email usage of the employees and may decide to investigate why person A sends 26 emails to person B each and every day. It can be used to measure performance of employees, comparing how fast they reply and how many replies they send to other employees. Then action could be taken to help or terminate employees who do not perform sufficiently.

The visualization tool can also be used for parties interested in air traffic. Users can plot all incoming and departing flights for major United States airports, comparing them and seeing the connections between them. Some interesting patterns can be found using for example the interleaved visualization technique that is described and implemented in the tool.

Programmers, both independent and corporate, could use the tool to visualize big programs. This way, they can keep track of function calls, functions that are related to each other, and possibly functions that are not being called at all. This way they could improve the efficiency of their programs when it comes to running times, and clean up their code so that it will not take as much storage space.

The visualization tool offers countless different ways to be used, different objectives to be accomplished, and has countless different users to be used by.

6 Discussion and Limitations

The visualization tool this paper is about has a few different visualization techniques, one of which has been optimized significantly since last time. The interleaved dynamic network graph has been optimized to be able to plot within a few seconds with the profile semantic trafo final dataset loaded in. This was a plain text file and was only 590 KB in size. Despite this fact, the radial and Fruchterman-Reingold plotting takes a long time to compute and load into the visualization tool. This can certainly be improved. In a previous version it was stated that due to computational limitations, the sliders with which the user can filter out certain time points or select a node range has predetermined steps to ease the load on the system. This has been improved significantly and the user is now able to adjust the slider to a node of his or her choice, it is however not clearly visible on which node the slider is standing, especially when a dataset that has a lot of nodes is loaded into the visualization tool. This is in violation of one of the principles of good interface design which is comprehensibility. This principle states that a system should be understandable, flowing in a meaningful and comprehensible order. Because the indications on the node range filter are depending on how many nodes are in a dataset. This is not always consistent and could confuse the users of the tool if they have multiple different datasets uploaded and ready for exploration.

Furthermore, the format of the accepted data is pre-specified and the designed and produced visualization tool will not work if the uploaded data file is not in the accepted formats (txt, csv.gz or dat_gz). The file loading system now checks for a few different formats of the input that were adapted to the specific datasets found on the internet. What could still be done is to let the user upload a file and ask what rows the time, weight, start, and target are. Then using this information, the tool will automatically modify the file to be in the correct format to be able to use it in the tool. Now it is possible that a file has the wrong start node which is actually the weight of the edge.

Together with improving the file loading support for other types of data such as tables, static networks, hierarchical data could be added. This way we would allow users to not only visualize dynamic graph network data. The user gets support for datasets that are structured in a different way, for example hierarchical data.

As of now we save much of data in memory to allow for fast interactions. This approach breaks however with the introduction of larger datasets, we would need to implement algorithms which are more space efficient. Another bottleneck is the amount of visual complexity which causes long plot rendering times. The interleaved plot had been improved a lot and can handle larger datasets. It started from 12 minutes on the profile semantic trafo dataset to 20 seconds. But the approach that is described above would help to let it scale to even bigger datasets.

Also, the node-link diagrams are pretty slow, especially with larger datasets than the one tested. Therefore, the responsiveness of these two plots, the Fruchterman-Reingold and radial layout, lacks. This can thus be improved in future work.

Since the previous version a stress test was executed on the tool to measure its performance in different fields, mainly the time it took for a plot to compute and how long it took for the matrix of the particular data to be plotted. To test this, a script was created that generates 10 different random dynamic graph datasets at different scales. These different scales include varying number of nodes, edges, and time range. Starting with about 20% of the size of the profile semantic trafo final dataset up to 300% of its size, at which point our web based visualization tool crashes. The console does still register a plotting time build into the code. Also, all underlying algorithms still work like they should but with the increased amount of plotting data that has to be loaded into the random access memory of the system causes the tool to 'crash'. However, it does not really crash. It is either that there is a shortage of RAM available or that the browser is just not able to render the complex visualization that is selected.

All the visualizations do work with the random generated datasets, but the node-link diagrams as well as the matrix look very scarce. This is because the data is generated randomly and thus they do not form many links between different 'clusters' of data, especially with small datasets.

So, the threshold of the visualization tool itself is at a dataset which is about 1800 KB in size. The code however, still works and registers the time it took to plot the graphs and the matrix. Because the tool itself crashed before the code, we assume this is related to the bad optimization of RAM allocation by our tool or just a shortage of RAM on the laptops of the developers. If the user has more RAM, the bigger datasets the visualization tool can handle. Therefore we do not know at what size of dataset the code will completely stop working as we do not have a system to test it on properly. The total time it took to plot the interleave dynamic network and the adjacency matrix against the size of the test file in KB can be found in Figure 9.

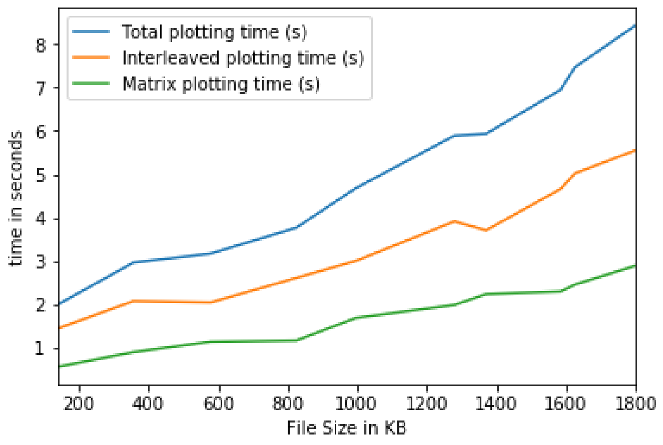


Figure 9: A performance chart for total plotting time, interleaved dynamic network plotting time and matrix plotting time against the size of the test file in KB

7 Conclusion and Future Work

In this paper we described our web-based, interactive visualization tool, that is based on an intuitive, chronological ordered graphical user interface that allows users to visually represent their data by three distinct visualization techniques - the radial-plot, Fruchterman-Reingold plot, and finally, an interleaved dynamic network visualization - to run a graphical problem on their data, in this case Dijkstra's shortest path algorithm, and to interact with it using the seven categories of interaction - select, explore, reconfigure, encode, abstract/elaborate, filter, and connect. To aid with this, a matrix overview of the data is available. This overview can be reordered as well. We explained these terms, the algorithms that were used, as well as several choices that were made in the creation of this tool. Finally, we explained how we envision the tool to be used and the format that the data should be in order to be used. For future work we plan to extend our work to make it applicable to bicluster data [21]. Also textual information in form of labels or word clouds might help to provide more semantic overviews [14, 26]. We should also focus more on the hierarchical aspects in the data [17, 18], for example, by clustering the vertices or time steps.

References

[1] M. Abdelaal, M. Hlawatsch, M. Burch, and D. Weiskopf. Clustering for stacked edge splatting. In *Proceedings of Vision, Modeling & Visualization, VMV*, pp. 127–134, 2018.

[2] F. Beck, M. Burch, and S. Diehl. Matching application requirements with dynamic graph visualization profiles. In *Proceedings of International Conference on Information Visualisation, IV*, pp. 11–18, 2013.

[3] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum*, 36(1):133–159, 2017.

[4] F. Beck, M. Burch, C. Vehlow, S. Diehl, and D. Weiskopf. Rapid serial visual presentation in dynamic graph visualization. In *2012 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC*, pp. 185–192, 2012.

[5] M. Behrisch, B. Bach, N. H. Riche, T. Schreck, and J. Fekete. Matrix reordering methods for table and network visualization. *Computer Graphics Forum*, 35(3):693–716, 2016.

[6] T. Blascheck, M. Burch, M. Raschke, and D. Weiskopf. Challenges and perspectives in big eye-movement data visual analytics. In *Proceedings of the 1st International Symposium on Big Data Visual Analytics*, pp. 17–24, 2015.

[7] T. Blascheck, K. Kurzhals, M. Raschke, M. Burch, D. Weiskopf, and T. Ertl. Visualization of eye tracking data: A taxonomy and survey. *Computer Graphics Forum*, 2017.

[8] V. Bruder, M. Hlawatsch, S. Frey, M. Burch, D. Weiskopf, and T. Ertl. Volume-based large dynamic graph analytics. In *22nd International Conference Information Visualisation, IV*, pp. 210–219, 2018.

[9] M. Burch. The dynamic call graph matrix. In *Proceedings of the 9th International Symposium on Visual Information Communication and Interaction, VINCI*, pp. 1–8, 2016.

[10] M. Burch. Property-driven dynamic call graph exploration. In *Proceedings of the 11th International Symposium on Visual Information Communication and Interaction, VINCI*, pp. 72–79, 2018.

[11] M. Burch, F. Bott, F. Beck, and S. Diehl. Cartesian vs. radial - A comparative evaluation of two visualization tools. In *Proceedings of 4th International Symposium on Advances in Visual Computing, ISVC*, pp. 151–160, 2008.

[12] M. Burch, M. Hlawatsch, and D. Weiskopf. Visualizing a sequence of a thousand graphs (or even more). *Computer Graphics Forum*, 36(3):261–271, 2017.

[13] M. Burch, M. Höferlin, and D. Weiskopf. Layered timeradartrees. In *Proceedings of the International Conference on Information Visualisation, IV*, pp. 18–25, 2011.

[14] M. Burch, S. Lohmann, F. Beck, N. Rodriguez, L. D. Silvestro, and D. Weiskopf. Radcloud: Visualizing multiple texts with merged word clouds. In *Proceedings of 18th International Conference on Information Visualisation, IV*, pp. 108–113, 2014.

[15] M. Burch, C. Müller, G. Reina, H. Schmauder, M. Greis, and D. Weiskopf. Visualizing dynamic call graphs. In *Proceedings of the Vision, Modeling, and Visualization Workshop VMV*, pp. 207–214, 2012.

[16] M. Burch, T. Munz, F. Beck, and D. Weiskopf. Visualizing work processes in software engineering with developer rivers. In *Proceedings of the 3rd IEEE Working Conference on Software Visualization, VISSOFT*, pp. 116–124, 2015.

[17] M. Burch, M. Raschke, and D. Weiskopf. Indented Pixel Tree Plots. In *Proceedings of International Symposium on Visual Computing*, pp. 338–349, 2010.

[18] M. Burch, H. Schmauder, and D. Weiskopf. Indented pixel tree browser for exploring huge hierarchies. In *Proceedings of the 7th International Symposium on Advances in Visual Computing, ISVC*, pp. 301–312, 2011.

[19] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Softw., Pract. Exper.*, 21(11):1129–1164, 1991.

[20] M. Greilich, M. Burch, and S. Diehl. Visualizing the evolution of compound digraphs with timearctrees. *Computer Graphics Forum*, 28(3):975–982, 2009.

[21] J. Heinrich, R. Seifert, M. Burch, and D. Weiskopf. Bicluster viewer: A visualization tool for analyzing gene expression data. In *Proceedings of 7th International Symposium on Advances in Visual Computing, ISVC*, pp. 641–652, 2011.

[22] M. Hlawatsch, M. Burch, F. Beck, J. Freire, C. T. Silva, and D. Weiskopf. Visualizing the evolution of module workflows. In *Proceedings of 19th International Conference on Information Visualisation, IV*, pp. 40–49, 2015.

[23] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letter*, 31(1):7–15, 1989.

[24] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. Visual analytics: Scope and challenges. In *Visual Data Mining - Theory, Techniques and Tools for Visual Analytics*, pp. 76–90. 2008.

[25] K. Kurzhals, M. Burch, T. Pfeiffer, and D. Weiskopf. Eye tracking in computer-based visualization. *Computing in Science and Engineering*, 17(5):64–71, 2015.

[26] S. Lohmann, F. Heimerl, F. Bopp, M. Burch, and T. Ertl. Concentri cloud: Word cloud visualization for multiple text documents. In *Proceedings of 19th International Conference on Information Visualisation, IV*, pp. 114–120, 2015.

[27] R. Netzel, M. Burch, and D. Weiskopf. Comparative eye tracking study on node-link visualizations of trajectories. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2221–2230, 2014.

[28] R. Netzel, M. Burch, and D. Weiskopf. Interactive scanpath-oriented annotation of fixations. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications, ETRA*, pp. 183–187, 2016.

[29] R. Netzel, M. Hlawatsch, M. Burch, S. Balakrishnan, H. Schmauder, and D. Weiskopf. An evaluation of visual search support in maps. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):421–430, 2017.

[30] R. Netzel, B. Ohlhausen, K. Kurzhals, R. Woods, M. Burch, and D. Weiskopf. User performance and reading strategies for metro maps: An eye tracking study. *Spatial Cognition & Computation*, 17(1-2):39–64, 2017.

[31] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pp. 336–343, 1996.

[32] C. Vehlow, M. Burch, H. Schmauder, and D. Weiskopf. Radial layered matrix visualization of dynamic graphs. In *17th International Conference on Information Visualisation, IV*, pp. 51–58, 2013.

[33] J. S. Yi, Y. ah Kang, J. T. Stasko, and J. A. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007.