Somewraps – Something with Graphs: Visual Analytics of Network Data

David Englert* Pablo Martinez Blasco† Eren Cakmak* Udo Schlegel* Juri Buchmüller‡
Daniel A. Keim§

University of Konstanz, Germany

Figure 1: Screenshot of `Somewraps`, that shows filter controls for the channels (A), controls for switching between views and layouts (B), the main view with node-link diagrams (C) and a brushable timeline containing daily activities per channel (D).

Abstract

We present `Somewraps` [1], a web-based visual analytics tool for the comparison of evolving network structures and their underlying labels. We elaborate how `Somewraps` can be used to answer the challenges provided in the VAST challenge 2020 dataset.

Index Terms: Human-centered computing—Visualization—Visualization techniques—Visual analytics.

1 Introduction

The VAST challenge of 2020 [1] introduces the fictional story of a global internet outage, caused by a group of white hat hackers. The challenge is separated into two parts. The primary goal of mini-challenge 1 is to identify the responsible group for the outage by comparing a given dynamic network structure against five potential matches. In the second part, the given template needs to be located within a large graph dataset.

In this paper, we describe our approach in solving the challenge by using the self-developed tool `Somewraps`, a web-based visual analytics tool that allows us to compare varying network structures. Additionally, the analysis is supplemented by other established network analysis libraries.

*e-mail: {firstname}.{lastname}@uni-konstanz.de
†e-mail: pablo.martinez-blasco@uni-konstanz.de
‡e-mail: juri.buchmueller@uni-konstanz.de
§e-mail: keim@uni-konstanz.de

[1] `https://vast20.dbvis.de/vast-mc1/`

2 Analysis

The challenge provides data of a large dynamic graph and subgraphs. Notably, we deal with various channels, e.g., email and phone and compare dynamic networks that contain time shifts. Due to the mentioned challenges and the overall complexity of the dataset, existing tools fell short. To tackle the described problems we designed and implemented `Somewraps`, Figure 1. It enables visual comparison of dynamic graphs with different views in an overview fashion with zooming and filtering options and details on demand. Additionally, it gives the possibility to explore specific parts in-depth, such as the travel channel with a custom visualization. The control elements (A), allow filtering by the different channels present in the dataset.

2.1 Embeddings

Graph comparison is often done using graph embeddings. Graph2vec [3] enables quantifying the similarities of the given graphs, however, it remains hard to compare and discriminate the given networks. After applying graph wave [2] for structural embeddings, clustered nodes already revealed more insights after encoding cluster information in the node-link diagram. We could identify specific roles in the network, such as main communicators and people with similar financial spending characteristics.

2.2 Node-link diagram

The node-link view, Figure 1 (C), allows a visual comparison of graphs, due to similar nodes being placed in the same area. This graph layout makes people with activities occurring solely in one channel or multiple people with similar activities, i.e., grouping alike behaving people, visually identifiable. For example, people with activity in the phone, email and travel channel are placed at the bottom left with a random layout.

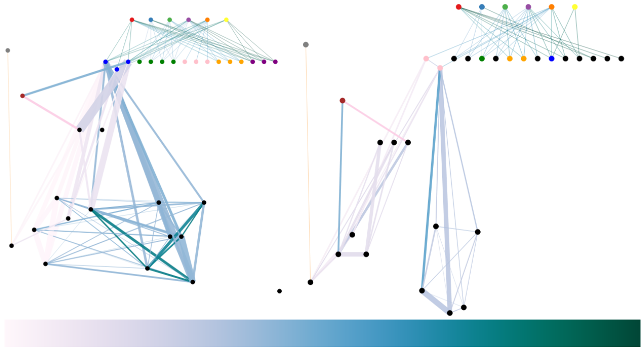

Figure 2: Node-link diagram with frequency encoded in the thickness of the edges and time in color. This reveals communication clusters in the time dimension and frequent communicators.

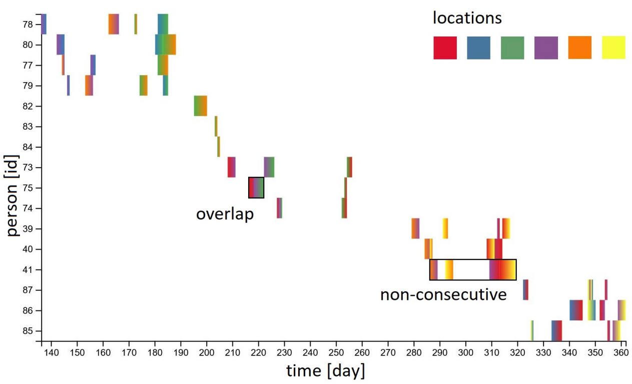

Figure 3: Travel activity with start, destination and duration information of a travel sequence represented as a rectangle with the respective length and location gradient.

2.3 Channel frequencies

By augmenting the node-link diagram with the encoding of time and frequency in the color and thickness of the edges, Figure 2, more patterns become visible. The phone and email channel can be split into three time clusters. Outliers, e.g., high frequent communicators, can be detected as well as balanced communication in groups.

2.4 Activity stream

The activity stream allows us to gain more insight into the activity of the channels. Figure 1 (D) shows the absolute aggregated amount of edges per day in each channel over the time range of the dataset. The view unveils significant patterns, e.g., high peaks or absence of any activity, and transactions that happen in each of the graphs in the temporal dimension. The stream allows us to narrow down the node-link diagram to a specific time range, which enables a more detailed comparison of the activity at a node level.

2.5 Travel timeline

`Somewraps` provides a novel representation for the travel channel. Each row in the travel timeline plot, Figure 3, represents a person's travel route, outlined by the given rectangles. Using gradients to represent the start and destination locations reveal groups of people who share common travel routes and times. Thereby, the visualization enables us to find consecutive as well as non-consecutive routes. Wrong gradients show travel routes that overlap and enable to identify data values that might be wrong or fake on purpose. Negative travel times are identifiable by black rectangles.

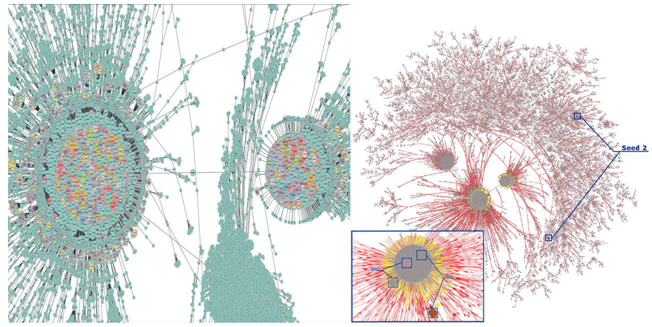

Figure 4: Drawings of the big graph by using a force directed layout: k-core subgraphs (left) and minimum spanning tree with seeds (right).

2.6 Big graph analysis

For the visualization of the big graph data, the python module `graph-tool`[2] was used in favor of `networkX`[3] due to computational limitations. As a plot of the big graph in a node-link diagram would be too vast, a force-directed MST representation of the graph, right Figure 4, was used. The nodes positions allow us to identify clusters, whereby three of them were recognized as being strikingly different from the rest of the graph.
The MST representation was used to locate the position of the given seeds, quickly ruling out seed two as it is visually discernible from its neighbours that it does not resemble the template graph at all.
A k-core decomposition, left Figure 4, of the graph was applied to find subgraphs whose k-core levels are close to the seeds and thus have the potential to be similar to the template graph. The decompositions enabled to rule out most of the graph and focus on two node clusters. One of them already contained the seeds.
Calculating an embedding of the big graph is computationally expensive, so the subgraph analysis was done manually based on features extracted from the template graph with the use of `Somewraps`. Based on this, seed one was identified as the most likely candidate leading to a subgraph similar to the template over seed three.

3 Conclusion

Our analysis of the dataset shows the necessity of custom visualization tools for complex data with multiple dimensions. `Somewraps` allowed us to encode domain-specific information in the layout of a node-link diagram that enables visual comparison of similar graphs. The interactive selection of prominent patterns in the activity stream allows a flexible comparison in the network dataset's temporal dimension within the patterns of interest. With the custom view for the travel channel, we gained an effective visualization with high expressiveness, that revealed in-depth details. The large graph analysis has shown the limits of using node-link diagrams and the high requirements of computational resources for big graph data. However, with the tool as an exploratory visual analytics tool, it was possible to abstract knowledge from the provided graph data by using knowledge extracted from the template network.

References

[1] K. Cook and R. J. Crouser. Vast challenge 2020. `https://vast-challenge.github.io/2020/index.html`, 2020.
[2] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec. Learning structural node embeddings via diffusion wavelets. 2018.
[3] A. Narayanan, C. Mahinthan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal. graph2vec: Learning distributed representations of graphs. 2017.

[2] https://graph-tool.skewed.de/
[3] https://networkx.github.io/