Joint Bachelor of Data Science
Bachelor End Project

Visualizing Dynamic Network structure by interactively slicing data on time-dimension

Yichen Wang
y.wang12@student.tue.nl

Supervisors:
prof. Roger Leenders
prof. Claudia Zucca

Eindhoven, June 2021

Acknowledgement

My deepest gratitude goes first and foremost to my supervisor, Professor Dr. Roger Leenders. It was a great honor for me to be part of the project he arranged. I sincerely appreciate him for pointing out the right direction for my work. Furthermore, the encouragement and support from him gave me the motivation and confidence to challenge myself in technical areas that I had never ventured into before and to gain knowledge from it.

Second, I would like to express my heartfelt gratitude to Professor Dr. Claudia Zucca for her constant encouragement in my work and valuable suggestions in my academic studies. Without her patient instruction, insightful criticism, and expert guidance, this thesis would not have been possible.

My sincere appreciation also goes to the teachers of the Data Science program from JADS, TU/e, and TiU, who have helped me develop the fundamental and essential academic competence.

Abstract

Continuous-time network visualization has been a challenging subject for years. It stems from the fact that the presentation of a relational structure of a network that consists of nodes and edges takes at least two dimensions, which leaves no adequate space for visualization to represent the information from the timeline. Although some approaches for continuous-time network visualization already exist, they are not very sophisticated. In this paper, we propose a dashboard-like visualization tool that gives the viewer an idea of the global trends of the development of network structure and local changes of entities and relationships that evolve. We apply our visualization tool to a representative time-series dataset of the epidemiology field, analyze the data through our approach, and give relevant recommendations.

Table of Contents

1. Introduction

In recent years, the study of networks and the corresponding network analysis has become a topic of great interest (Moody, McFarland & Bender-DeMoll, 2005). Every entity, whether concrete or abstract, is connected to others by relationships and constitutes network. By depicting the relationships among entities and analyzing the structures that arise from the recurrence of the relationships, network analysis can be applied to a variety of fields such as social sciences, communication behavior, life sciences to help us with a deeper understanding of relationships and provide better explanations of social phenomena (Chiesi, 2001).

Since Sociograms were introduced by Moreno (Moreno, 1953), visualization has long been an essential tool for analyzing social network structure. Typically, the relevant research data was collected at a particular point in time. Researchers used nodes and edges to represent objects and relationships with a static structure in which nodes never crash and edges maintain operational status forever. However, social networks evolve asynchronously; it has been shown that the structure of the network, as well as the characteristic features of entities, may not only mutually depend at a certain point in time but also be influenced from the past in a continuous-time dimension (Snijders, 2005). Thus, if we would like to observe the change and properties of the network over time, a continuous-time dimension would be introduced into the network, in which data is continuously observed and collected. Such continuous-time networks have dynamic structures that vary over time which nodes may come and go, and edges may crash and recover (Rajaraman, 2006).

Nevertheless, continuous-time network visualization has been a challenging subject for years; developers only have progress on optimal network layout, such as *multidimensional scaling* and *singular value decomposition* (Freeman, 2000; Brandes, Raab & Wagner, 2001) but are struggling with how to present the time-based connected data with an intuitive way to expose the information which folded in the timeline (Shi, Wang & Wen, 2011). It stems from the fact that the presentation of a relational structure of a network that consists of nodes and edges takes at least two dimensions, which leaves no adequate space for visualization to represent the information from the timeline (Moody et al., 2005). On the other hand, suppose using one axis represents time, then only one axis remains to convey the network topology. Although some trends can be identified on the one-dimensional time series diagram constructed in this way, it can not represent the network structure well. Although some generic tools for dynamic network visualization have already existed (Liu et al., 2015; Shi et al., 2015; Everton, 2004), they are not very sophisticated. A brief review of some representative early work and approaches will be expounded in the next section about their pros and cons.

Furthermore, the process of a network analysis consists of a sequence of diverse tasks from various aspects (Hadlak, Schulz & Schumann, 2011). For example, a user may first need to recognize patterns or select certain intervals through an overview and then zoom in to a specific continuous time range to analyze the structure further. As a result, the user usually cannot choose a single visualization once and for all but must instead switch between various visualizations to always use the best one for a particular phase of analysis. This is also the case if the user selects different subsets of the data with different characteristics requiring particular visualization, such as switching the structure of a diagram to a spanning tree when necessary to focus on one entity for egocentric network analysis. Whereas most existing approaches focus solely on individual visualization layout and techniques, this paper is concerned with strategies to combine multiple diagrams or interactively shift between them using an integration mechanism.

In addition, from social and biological processes to financial and communication systems, each industry has its specific needs and segments for the research of the network and the functions of the corresponding visualization tools in a professional field. However, those generic approaches do not carefully examine the meaning and implications of time in the formation of networks, and the tools are not tailored to specific domains or meet the needs at the application level.

Based on the above, in order to visualize a continuous-time network, the research direction and development objectives in this paper that need to be approached simultaneously can be summarized from three aspects:

1 How to display the trend and summary statistics of the entire network in time series.

2 How to display the changes in the time dimension of the structure composed of entities and relationships.

3 How to meet the requirements of continuous-time network analysis in the professional field in terms of functionalities.

Therefore, we propose a requirements-based method to solve the problem of visualizing continuous-time networks. This method starts from analyzing the characteristics of a representative dataset and the needs of its related research area, examining carefully questions about the meaning of time in forming networks, and ensuring the data are processed appropriately. Next, a designed data structure is demonstrated to show the underlying logic of data storage and manipulation. Afterward, some critical algorithms and implementations of graph theory and social network analysis are introduced that justify the type of questions that can be answered by our approach, which meet the needs of continuous-time network research for both the summary statistics over time and the changes of the network structure in the selected time range. Finally, the architecture of the application and GUI design are presented to show how interactive operation proceeded between multiple visualization components.

2. Literature Review

There are two main challenges that need to be considered in order to visualize temporal networks: *graph structure* and *temporal domain* (Hadlak et al., 2011). The former focuses on accurately describing the network's relationships between entities and the features they contain. In real-world data, however, the relationships between instances can be very dense, leading to interlocking and overlapping internal structures of the graph, making it difficult to read. At the same time, entities and relationships may contain multiple features with various values, whereas it is difficult to unfold as many features as possible at the same time or to switch features on the fly according to user needs. The *temporal domain* refers to how data is deconstructed and presented in the temporal dimension. It is generally accepted that the temporal dimension, which is encoded in the network structures, is categorized into two distinct forms: discrete and continuous (Moody et al., 2005). The discrete-time networks are constructed and presented based on each point in time, and each discrete time can be viewed as a cross-sectional snapshot in the time dimension. Thus, discrete-time networks focus on changes in the state of the network structure at different points in time without reference to the relationship and influence of the time series on the changes. The continuous-time network contains the explicit time of existence and dies out of entities and relationships, and it expresses how dynamic events affect network change through sequentially ordered. Thus, unlike discrete points in time, continuous-time encompasses the concept of a time span for which data is available in certain units of time, and the units may also be fluid.

On the other hand, discrete-time and continuous-time are to some extent interconvertible with each other. That is, continuous-time can be split into a nearly infinite number of discrete-time, and conversely, multiple discrete-times of the same time interval can approximate continuous-time. Besides, the size of a single static network varies positively and linearly with the number of instances and relationships it contains, whereas the size of a dynamic network containing a temporal dimension grows polynomially with time span. Therefore, it is crucial to choose the correct time representation and define the temporal boundaries of the network. The existing approaches of time-based network visualization are consequently achieved by either reducing structure, reducing time, or both. However, all of these approaches have more or less certain drawbacks.

The reduced structure visualization generally preserves the complete timeline of the original data, but only part of the network is selected in specific interests. One of the representative approaches is Fig.2.1, which displays and analyzes the dynamic network from the egocentric view of a specific entity. The double-sided temporal glyph placed vertically exposes the entity's existence, and the width of the glyph on the unit interval represents the total number of relationships associated with the entity. The entire visualization is visually divided into two areas by this glyph and given different functionalities. The ribbon on the right shows a further illustration of all other entities and relationships associated with the egocentric entity under a cross-sectional snapshot of the timeline when continuous-time is

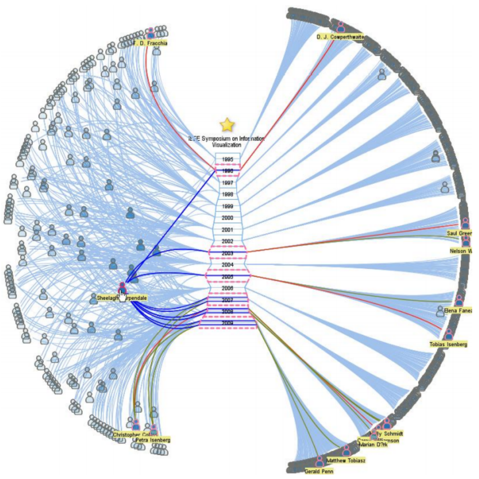

Figure 2.1: Dynamic Network Visualization in 1.5D (Shi et al., 2011)

converted to discrete-time.

In comparison, the ribbon on the left shows more clearly the period of each relationship between the egocentric entity and the other. Thus, the natural clustering of a specific entity could be displayed by both discrete and continuous-time views in such visualization (Shi et al., 2011). However, there is no denying that the reduced structure visualizations partially reduce the clustering, resulting in a large amount of information of the essence being lost. For example, the visualization only provides the degree and the number of adjacent entities of the egocentric entity. In contrast, there is no information about the overall scale or size of the clustering in which the entity is located, thus causing the user to make incorrect cluster analyses. In addition, if the time span of the dataset increases or the duration of the relationships increases, the layout of the complete visualization becomes very crowded, with nodes and arches overlapping in layers, making it impossible to distinguish between them directly to the naked eye.

The reduced time visualization preserves the complete network structure with all active entities and relationships on a certain cropped period of continuous-time or a sequential series of discrete times from the timeline. Fig.2.2 is an approach by network analysis software *Pajek*. Thus, the data is sliced equally into discrete-time, and multiple consecutive discrete times are selected to represent a continuous period. In each discrete-time, the graph structure with its entities and relationships is unfolded completely, while a linear transformation compresses its layout to place more consecutive discrete networks simultaneously in the same space. Moreover, additional edges are generated to concatenate the same entities throughout different time slices. It could point out if the corresponding entity and relationship exist or die out.

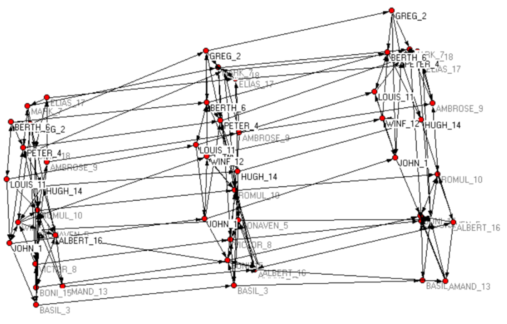

Figure 2.2: Sampson Liking Data at Three Points in Time by Pajek (Everton, 2004)

Nevertheless, the time-reduced approaches require a certain level of background knowledge and reading ability from users due to the extra elements involved. Otherwise, users can not compare networks in different time slices and get informative results. In addition, as the number of selected time slices increases, the total number of elements that need to be displayed grows exponentially, causing each network to be overly squeezed and distorted beyond reading. Thus, the reduced time visualization approaches limit the scope of data selection, allowing users to observe only changes in dynamic networks over adjacent short periods.

To overcome the disadvantages mentioned above and to enable the complete network structure to be presented over the entire timeline, instead of static visualization, using the animation approach to display network movies by temporal views was applied in some continuous-time network studies and some well-known visualization libraries, such as *ndtv*, *vignette igraph*, they already could either provide animation features or call the interface of *animation* library to achieve such function (Fig.2.3). However, the animation approach has been questioned that it may not be very effective for network analysis tasks. On the one hand, some experiments revealed that animated visualization performs worse in terms of readability than static image counterparts. When animation switches between sequential frames, some extra interpolated frames are generated to achieve a smooth transition, and these extra interpolated frames can somewhat interfere with reading and prolong the reading time. Changes in the network between two keyframes of data slices, on the other hand, must be observed by comparison without any highlight, requiring users to frequently switch between keyframes, which reduces the user experience. Another serious drawback of the animation approach is that each keyframe can only represent a discrete-time, so the time span of the existence of entities and relationships is not well-reflected. Therefore, the result of this visualization is not really "continuous-time".

Considering that the start and end time of the existence of entities and relationships are essential features of a continuous-time network, Bernder-deMoll and McFarland proposed the concept of *sliding window* that spans a set of sequentially relational events. Entities and relations whose existence falls entirely within that interval or that are ongoing beyond the boundaries of the interval are included in the network (Bender-Demoll & Mcfarland, 2002). Within the *sliding window*, the existence of entities and relationships may be temporally sparse, so the inevitable question arises as to how the different elements within the window over many periods can be aggregated to produce a meaningful image of the network structure. One effective method of displaying the sparse network is proposed to show how the network emerges over time by dynamically adding the entities and relations when they appear and placing them in the final aggregated structure (Moody et al., 2005). This operation of accumulating, together with an animated demonstration, can effectively reflect the development of the network and the expansion of its size.

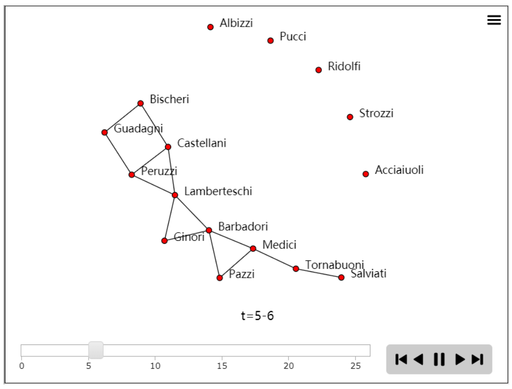

Figure 2.3: Animation network with *ndtv* by R. (Ognyanova, 2017)

From the above review of the literature and existing approaches, we recognize that the main topics to be addressed in the development of this application include the following:

- Determine the appropriate form to encode temporal dimension based on the format of raw data.
- Determine the method of data reduction
- Implement an effective merging operation for *sliding window*

In addition, other literature and theoretical support on data structures, algorithms, and UI/UX design will be discussed in the development process methodology.

3. Dataset

3.1 Background

The dataset chosen for our application development was collected from a primary school in Lyon, France. Ten classes with a total of 232 children and 10 teachers were observed as participants in the experiment in school activity over two days (1^{st} Oct and 2^{nd} Oct) during the 2009 H1N1 pandemic. All participants wore the wearable device with a sensor that could detect a face-to-face proximity relation by sensing an RFID chip in other wearable devices with a 20-seconds temporal resolution (Cattuto et al., 2010). The dataset contains two files, one with a total of 125,772 contact events, each event containing the unique id of a pair of individuals and the ending time of a 20-seconds resolution sensoring. The other file contains the characteristics of each individual, including unique id, class, and gender.

The behavioral patterns of students and educational practitioners within a school is an important topic for medical research, as students and teachers are a crucial epidemiological group for transmission of influenza-link viruses (Longini, Koopman, Monto & Fox, 1982). This is due to the limited prior immunity of students, especially in the early grades, who tend to be more susceptible to viruses. Furthermore, the schedule of activities within schools results in a high rate of contact between students and is, therefore, a driving force for the spread of the virus (Mossong et al., 2008). Additionally, children play a very pivotal role in spreading infectious diseases in the community, as children need to be cared for by parents or grandparents outside of school hours, which can accelerate the spread of the virus in the form of a cross-cluster (Willis et al., 2019).

3.2 Objectives

During the 2009 H1N1 epidemic, many countries took measures to close schools as a way of breaking the vital chain of transmission to reduce the spread of infectious diseases in the community, in order to delay the peak and minimize the impression of an epidemic, and there were reports that the epidemic was brought under control to some extent as a result (Huang, Lipsitch, Shaman & Goldstein, 2014). However, school closures are undeniably associated with high socio-economic costs, as parents have to take care of children and may be forced to take time off work, adversely affecting the normal functioning of society. Researchers analyzing this continuous-time data aim to assess the mode and severity of transmission of infectious diseases in schools and to provide guidance on how schools can react to the risk through partial closures while minimizing the socio-economic impact of strict epidemic prevention policies (Gemmetto, Barrat & Cattuto, 2014).

The objectives of the study based on this data and the results of its research are of great academic importance and practical importance. It is considering that the world is still experiencing the global pandemic of Covid-19 and that the debate on whether schools should be closed when the number of infections rises is still inconclusive. Therefore, we have developed an integrated visualization tool that can be applied to a continuous-time network for medical and epidemiological research. Our tool shows a clear picture of dynamic network changes from both discrete and continuous-time perspectives while providing reference indicators needed for epidemiological purposes, such as the close contact groups, the contact density of functional relationships, the central intermediaries along the transmission pathway.

4. Methodology

This section describes the corresponding methodologies according to the sequence of the application design and development process. The applied data structure is discussed in the first part. We nominate the candidate data structures based on the raw data format and then compare storage space and computational complexity to determine the final adopted solution. In the second part, we describe the implemented measures and algorithms and justify their practical use in the continuous-time network analysis, particularly in epidemiological research. In the third part, we illustrate the application's architecture and its running platform, describing in detail the advantages of our choice. The last part relates to the UI/UX design. From optimizing user experience, we planned the application interface in terms of visual perception and interaction logic.

4.1 Data Structure

Choosing and adapting an appropriate data structure is the basis of our application development. As described in the previous section, the raw data consists of two parts: *total contact events* and *features of individuals.* The measurement of contact events is based on a resolution of 20-seconds. Thus a discrete network is generated every 20-second. Schiller proposed an efficient data structure specifically for dynamic networks with long time spans. Its properties include a base network structure G_0 at time point T_0, and changes at any other time point T_i concerning the previous time point T_{i-1}, i.e., which relationships have been added or extinguished. Therefore, G_i could be derived from G_{i-1} by constant runtime (Schiller, Deusser, Castrillon & Strufe, 2016). The advantage of this data structure is it takes less storage and space complexity. It also has the obvious disadvantage that the data needs to be recalculated from T_0 with $O(n)$ time complexity when the network of a particular point in time or time period needs to be displayed. Whereas during data exploration, the user may often be browsing visualization from various time periods and comparing them before and after, leading to a tremendously high computation. Hence, we decide to use a *hash table* as the main data structure, with the time of the recorded event as the key, mapped to the corresponding discrete network. Thus, it takes $O(1)$ on selecting discrete network and implement the *sliding window* on a particular period of continuous-time from T_i to T_{i+k} with $O(k)$ complexity, where k is far less than n.

For each individual discrete network, we use *graph*, a non-linear data structure, to represent entities and relationships in terms of *vertices* and *edges.* Denote graph as $G = (V, E)$. In practice, there are two ways of presentation of a graph: *adjacency list* and *adjacency matrix*, and they have their specific advantages over each other in some respects. Therefore, for the choice between the two, the following factors were considered based on the characteristics of the raw data.

Storage Space: For a dataset containing time series, the larger the time span, or the greater the accuracy per unit of measurement time, the more storage space it requires. The dataset we have chosen contains 3,100 measurement times, which means that this continuous-time network is transformed into 3,100 graphs, so minimizing storage space is a primary consideration. Through data exploration, we observed that the average number of entities(vertices) with contact records in the discrete network was 56.39, which means that only 23.3% of individuals were active compared to the total number of observations. Moreover, the average number of relationships(edges) is 40.57, reflecting a low average actual density ($\frac{2|E|}{|V|(|V|-1)} = 0.026$) and sparse structure of proximity contact between students in campus activities. Therefore, the advantage of the *adjacency list* ($O(|V|^2)$) over the *adjacency matrix* ($O(|V| + |E|)$) in terms of storage space is particularly important.

Computational Complexity: Our application implements the *sliding window* by performing merging graphs on a series of consecutive discrete networks. To perform a merge operation on any two graphs, it is required to iterate over one graph and then adding or removing vertices and edges to or from the other graph. Thus, total computational complexity of adding for *adjacency list* takes $O(|V|+|E|) \cdot (O(1)+O(1)) = O(|V|+|E|)$, whereas *adjacency matrix* takes $O(|V|^2) \cdot (O(|V|^2) + O(1)) = O(|V|^4)$, removing takes $O(|V|+|E|) \cdot (O(|V|+|E|)+O(|E|)) = O((|V|+|E|)^2)$ and $O(|V|^2) \cdot (O(|V|^2)+O(1)) = O(|V|^4)$ respectively.

Hence, the adjacency list was finally adopted as the data structure for storing sparse networks and applied in our approach.

4.2 Algorithms

The main goal of our application is to achieve both converting a given series of discrete-time networks to a continuous-time network and measuring the given network by using appropriate algorithms. As previously mentioned, more extended time spans or shorter time intervals can cause the size of a time-series-based dataset to increase dramatically, so that the sliced data from *sliding window* may also require high computation cost. Therefore, we optimized the algorithms to reduce the running time of the implementation and did some brief benchmark tests to validate the improvement experimentally.

4.2.1 Network Converting

The *sliding window* proposed by Bernder-deMoll and McFarland defines an interval (from "slice start" to "slice end") and then aggregates or superimposes elements that fall inside the interval seriatim from slice start to slice end (Bender-Demoll & Mcfarland, 2002, Moody et al., 2005). On the basis of this concept, we implement the approach that for a given pair of time T_i and T_{i+k}, merge the sequence of graphs $G = \{G_i...G_{i+k}\}$. Therefore, we introduced three merging methods into our application, including: *Union*, *Intersection* and *Difference*. Each method corresponds to a specific usage scenario for different needs of research objective.

Union: The main interest of the continuous network analysis is to observe the structure of the active entities and active relationships that exist over a continuous period and the duration of their existence. This, therefore, requires that the merged result retains all elements within the *sliding window* as completely as possible, including the features of the entities and the weights of the relationships. Thus, we define the union operation for merging graphs such that $G = (G_i \cup ... \cup G_{i+k}) = (V_i \cup ... \cup V_{i+k}, E_i \cup ... \cup E_{i+k})$.

Intersection: On the other hand, to define a temporal boundary for a continuous-time network in practice may also be in terms of the existence of events. For example, to find a continuous-time from T_i and T_{i+k} such that entities or relationships exist throughout.

In such a case, although the temporal boundary can be obtained by adjusting the time span of *sliding window* and looking for entities or relationships with full weights in the union graph, it has certain drawbacks due to the objective of the analysis changes, not all elements from the union graph are getting interests. These redundant elements cause unnecessary additional computation costs and distract the user when analyzing the network structure. Consequently, we introduce the intersection operation according to the mentioned need such that $G' = (V_i \cap ... \cap V_{i+k}, E_i \cap ... \cap E_{i+k})$.

Difference: Other than realizing the existence of all elements or the relational structure of the network within a certain period. The endogenous network processes are often getting interested by researchers (Moody et al., 2005). Examples of our selected dataset include identifying how the newly active teacher transforms the teaching environment from a clamor to a well-organized one. The cross-class and cross-grade relationships are increasing dramatically because some students are perking up. In such cases, the structure of a discrete-time or continuous-time network G with time span $T_{i,j}$ may be influenced by the structure at time T_i^-. It could also influence the succeeding structure at time T_j^+, and the goal of visualization is to point out the newly raised entities and relationships which exist in G but not G^-, or the dying out entities and relationships which exist in G but not in G^+. We introduced the difference operation, that for given two graphs, say G_i and G_j, the difference graph $G' = G_i - G_j = (V_i \setminus V_j, E_i \setminus E_j)$. Therefore, the new raised equals $G - G^-$, and the dying out equals $G - G^+$.

Although the theoretical complexity of both union and intersection operation on any given two graph, say G_i and G_j, are the same with $O(min\{V_i, V_j\} + min\{E_i, E_j\})$. To a sparse network of time series data in the real world, since the intersection is a sieve operation on elements, it takes much less space complexity and running time as the number of iterations increases than for union operation. Fig.4.1 is a benchmark testing on the performance of union and intersection operations. we took a *sliding window* of 10-minute time span, which contains 30 discrete-time networks from the school activity dataset, and take both merging operations independently execute 200 times. After the results of running time were collected and being statistically analyzed, as the data of running time is non-normally distributed, a one-sided Mann-Whitney-U test was applied for the hypothesis testing with a p-value of 5.053×10^{-68}, shows that the distributions of time-consuming of two operations are not equal, and union operation takes significantly more time (3.334 milliseconds by average) than intersection (0.494 milliseconds). The result confirms that intersection graphs can be an efficient alternative to union graphs where the user's needs can be met.

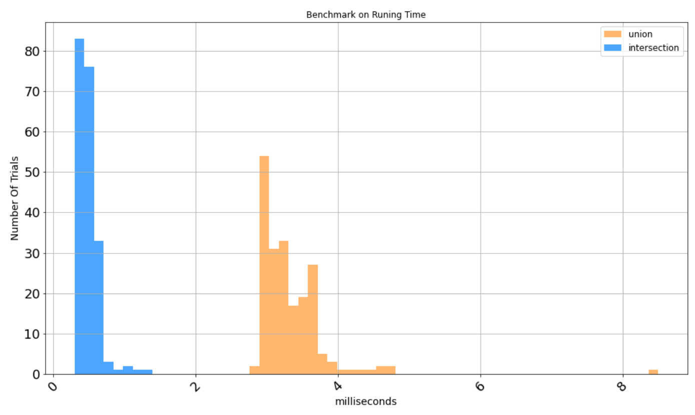

Figure 4.1: Distribution of running time by benchmark testing of union graphs and intersection graphs through 200 times trial

4.2.2 Network Measures

Social network metrics are wildly used to study disease ecology and epidemiology(Silk et al., 2017). This is because the interaction between individuals is one of the primary means of infection spreads, and therefore the structure of social networks is an essential subject of epidemiological research (Newman, 2002). Social network analysis measures could be usefully applied to quantify the behavior patterns at the individual-level and population-level. Some specific measures are introduced into our application for revealing the patterns at both levels.

Individual level The position of individuals in a social network reflects the role that the individual plays in the transmission of infectious. For example, individuals who frequently interact with large numbers of people could be considered as super-spreaders (Lloyd-Smith, Schreiber, Kopp & Getz, 2005). In contrast, individuals who interact across groups may act as bridges between different parts of the network (Weber et al., 2013). The following measures are included in our application with the practical implications of these measures in epidemiological studies.

Degree of vertex in a social network is the number of connections the node has to other vertices. If the relationship between entities is directed, the degree of vertex could be classified into in-degree and out-degree. Individuals with high out-degree are more likely to spread the infection to others and could be considered super-spreaders, while individuals with high in-degree are more likely to be exposed to infection.

Weight of edge in our designed continuous-time network represents the length of time of a relationship that exists within a continuous-time. Edge with high weight refers to more potentially contagious. The duration of interaction between two individuals determines per-contact transmission probability, primarily when diseases are transmitted by droplets or physical contact. A prolonged contact or unprotected exposure leads to a higher likelihood of disease infection. (Smieszek, 2009).

Node betweenness centrality determines how often a node's extent of lying in the connective path between two other nodes. Node with high betweenness centrality connects more parts of the network and gains greater internal network influence (Bloodgood, Hornsby, Rutherford & McFarland, 2017). It helps to find which entity takes the most critical role to mediate the spread of infection, thus helping researchers to be able to effectively interrupt the spread of infection by controlling interactions of the mediation. The implementation of our algorithm first calls Dijkstra's algorithm with min-priority queue to find the shortest path for each pair of vertices in the network, then compute the unnormalized metrics according to the expression: $g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$ (Freeman, 1977), which take a worst-case runtime complexity of $O(|V|^4)$.

Population level Some metrics can be applied to describe the susceptibility of the parts or whole network to infection, as well as at which epidemics may spread through it (Silk et al., 2017). They are particularly useful to the detection of substructure in combination with the measures of individual level. For example, when the overall structure of the network is community-distributed and, infected hosts are more clustered, and if there are fewer bridge-like entities, we could determine that infection will spread relatively slowly (Lloyd-Smith et al., 2005). Employed Measures are listed below.

Disconnected subgraphs represent unconnected subdivisions within the overall network. It refers to a naturally blocked transmission of infections. Therefore, a network with more disconnected subgraphs means a relatively small range of infection so that the spread of disease can be more easily controlled. Our approach achieves a worst-case runtime complexity of $O(|V|^2)$ by cutting nodes according to the minimum spanning tree based on the depth-first search algorithm (Bondy & Murty, 1976).

Edge betweenness centrality is extended from node betweenness that defines the number of the shortest paths that go through an edge in a network. The property of community structure in a may network was highlighted by Girvan and Newman, in which the nodes of the network are linked into close-knit groups, between which there are looser connections (Girvan & Newman, 2002). In epidemiological studies, for example, using edge betweenness centrality as the method to identify the natural communities in a dense social network and accentuate the behavioral pattern and featured pattern on why some nodes are tightly joined.

Density describes the proportion of potential relationships in a network that is actually existed. In daily school activities, an increase in network density represents that individuals are getting closely aggregated. Considering infection transmission would be expected to spread more rapidly in a dense network as the social distancing may not be obeyed. The density of the network is an effective metric for monitoring students' behavior or measuring teachers' teaching methods.

Maximal cliques in graph theory describes a close-knit group in a network that for every two distinct individuals in the group, there exists a relationship. In contrast to density, clique tends to show the local state of the partial network while comparing a sub-structure with the rest of the network and analyzing the similarities and differences between them. However, to determine if a clique of a given size in a graph is NP-complete (Bondy & Murty, 1976). Thus, for finding all maximal cliques in a given network, the Bron-Kerbosch algorithm (Bron & Kerbosch, 1973) is introduced to our implementation.

The advantage of using Bron-kerbosch algorithm for the clique problem is the high efficiency in a worst-case sense with runtime complexity $O(3^{\frac{|V|}{3}})$ (Bron & Kerbosch, 1973), where it has been proved that the maximum number of distinct maximal cliques is $3^{\frac{|V|}{3}}$ (Moon & Moser, 1965). In the early stages of development, we first implemented the classic Bron-Kerbosch algorithm, which uses a recursive backtracking method for every clique and identifies if maximal or not. However, late in the development, we found this version of the algorithm is inefficient due to its searches and tests all vertices in order, whereas the neighbor of tested vertex should not be necessary to test. Therefore, we adapted the algorithm with pivoting approach (Bondy & Murty, 1976) to save time and allow backtracking

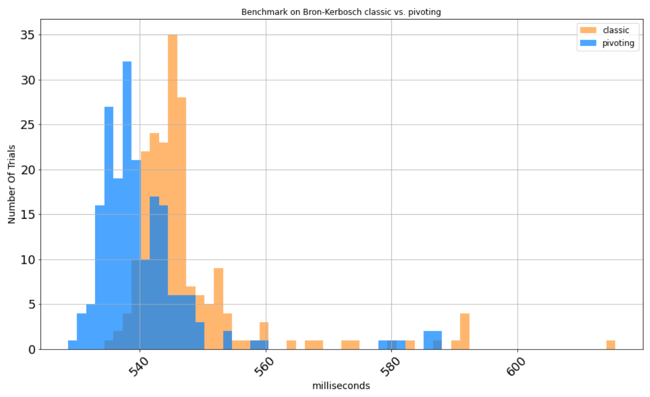

Figure 4.2: Distribution of running time by benchmark testing of classic and pivoting Bron-Kerbosch algorithm through 200 times trial

effectively in branches of the search and test. A benchmark testing on the performance of both versions of Bron-Kerbosch algorithm was implemented (Fig.4.2). We execute 200 times running both algorithms on 3100 discrete-time networks. The mean running time of classic Bron-Kerbosch algorithm takes 547.856 milliseconds, and pivoting version takes 540.818 milliseconds. A one-sided Mann-Whitney-U test was applied for the hypothesis testing with a p-value of 1.299×10^{-27}, which shows the optimized algorithm with pivoting approach performs significantly better than the classic version.

4.3 Application Architecture

Unlike visualization libraries that only support a single programming language or visualization applications that require pre-installation, our visualization tool is a web application developed in a mixture of programming languages based on the Flask framework. This choice was based on the following reasons:

No Prerequisites: Our application encapsulates all the code in simple click-and-drag operations so that the user can visualize and manipulate the uploaded data without any programming basis. This allows users to explore the data directly through visualization without additional learning costs.

Easy Access: The different operating systems and devices generally limit the visualization software that needs to be pre-installed; otherwise, separate versions have to be developed for the different platforms. However, a web application is an excellent solution to this problem, requiring only a browser and allowing any platform and device to access the server for data analysis and visualization. It increases flexibility and scalability. At the same time, Flask, as a lightweight web framework, makes it easy to set up a local server to use our open-source application, even when offline.

High efficiency and expansibility: Flask is a web application framework written in python (Flask, 2010), so its back-end server-side can directly call python scripts and use pandas, NumPy, SciPy, and other open-source libraries, which are widely used for data analysis to process the uploaded data, but also the script generated static visualization image could be rendered directly to the front page. In addition, our application uses asynchronous functions to initiate HTTP requests for front and back end interaction and data form submission, and the URL dispatcher of Flask is a RESTful request. This feature is a great help in developing and managing the API comparing to other web frameworks written in python, such as Django, CherryPy. Moreover, Flask supports multiple databases and is advantageous for later application development, such as data storage and migration.

Fig.4.3 shows the logical architecture of the application. The user first selects and uploads the files on features of entities and time series-based relationships that need to be visualized and analyzed while declaring the type of delimiter (comma, semicolon, or tab-separated) and header. After these files and parameters have been validated (file type, size, etc.), they will be serialized and posted to the server via a new asynchronous process. The server side will store the files in the temporary folder, load and read the first 5 lines of data by a python script, and then encode them into Json format and return them to the front end so that the user can preview the data content. Afterward, the user needs to assign the feature names on the previewed data. The mandatory features for relationship data are "time", "id" of entity i and j, and the mandatory feature for entities data is "id" to ensure that these two files can be accurately associated. The server subsequently reads and pre-processes the entire dataset based on the validated parameters of the new submission and returns the jsonified dataset to the front end. Ultimately, the JS script is called on the browser end to store the complete dataset into the designed data structure and to perform calculations based on the user's requirements and present them visually by visualization.

Our application architecture is designed base on the separation of front-end and back-end. All algorithms and computations are implemented on the front-end via the browser engine. The back-end server is only responsible for the data transmission and the pre-processing of a raw dataset, while the temporary storage of raw data storage is separated as well. This design solution allows our applications to be more easily extended and maintained. By only modifying the corresponding front-end module, our application can be rapidly iterated in case the components of the visualization tool need to be updated or the method of data analysis needs to be changed. In addition, when updating the raw dataset, the server only pre-processes the data separately without calculations or amendments, and there is no need to re-render the page. At the same time, asynchronous HTTP requests and data transmission between the front-end and back-end can ensure the application process not be blocked. This improves server performance a brings a better user experience (Joshi, 2018).

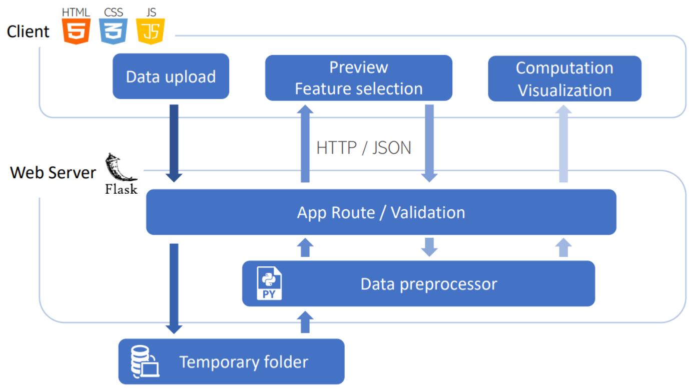

Figure 4.3: Architecture of web application

4.4 UI/UX design

Our application uses Bootstrap, the open-source front-end toolkit released by Twitter, to styling the various visual components and controls, making the page look more coherent overall. Also, in order to provide a good user experience, our application is divided into five functional areas according to the sequential logic of the basic approach of data exploration:

1. data upload
2. time-based summary statistics
3. sliding window selector
4. node-edge network
5. statistics of network

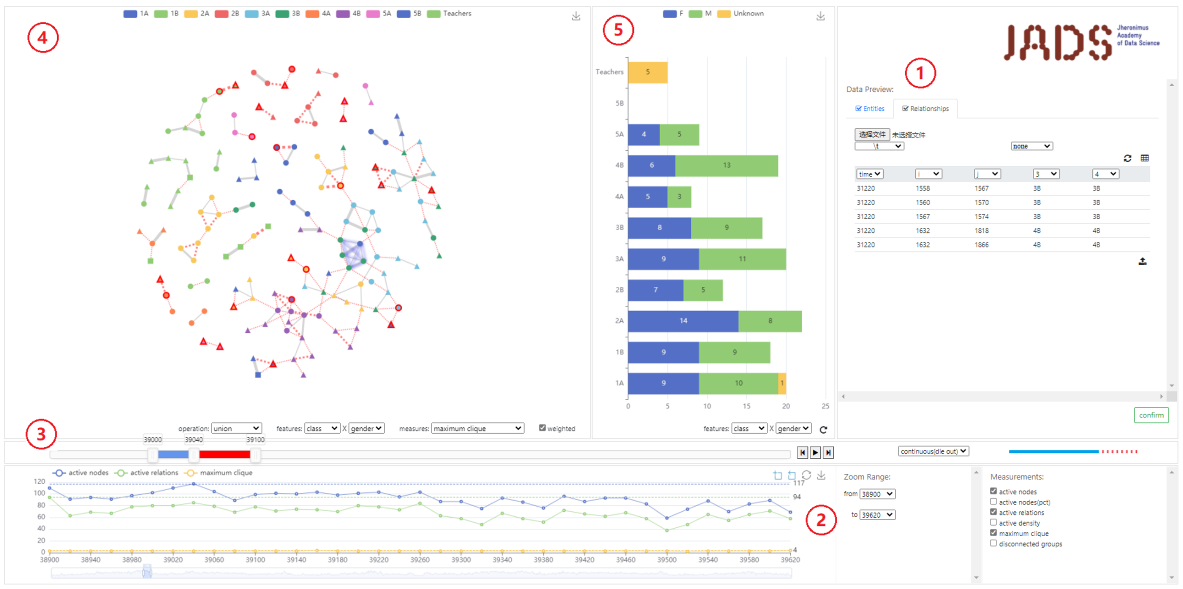

Figure 4.4: Graphical User Interface

These functional areas are organized in an adaptive grid based layout which could fit a variety of screen sizes (Fig.4.4). Furthermore, the design for the placement and order of areas is based on the following considerations:

4.4.1 Screen Hotspots

The visualization of the dynamic network structure is an essential part of our application. Moreover, the observation and analysis of the network structure is also the most critical and time-consuming part of the study of continuous-time networks. Therefore, the position of visualizations needs to be adapted to the user's habits. According to a user experience study by the Nielsen Norman Group, web users spend much more time viewing the left half of a page than the right half, and the gap between the two is increasing year on year, with the left half of the page already being viewed 80% of the time in 2017. This also means that designing pages to follow this behavior will significantly improve user efficiency (Fessenden, 2017). Also, multiple studies have shown that the top-left zone of the web page gets the highest priority of attention (Pernice, 2014; Pan et al., 2004). We have managed all the visual components on the left half of the page, especially placed the node-edge graph in the top-left corner and occupying 37.5% of the page, the ratio of 1 : 1.67 to the area of the remaining part, which approximates to the golden ratio. Therefore, the main interest is positioned in the preferable visual area to focus more on reading and analyzing the visualization, thus improving the user experience.

4.4.2 Visual Flow

To enable users to quickly understand and use the application without the need for textual explanations, we direct the user's attention along a given path by adding the visual weight of elements and adjusting the layout according to the approach of visual flow(Pang, Cao, Lau & Chan, 2016). We distributed the five functional areas clockwise based on the logical order of data exploration operations, placing the less frequent and less time-consuming functional areas or functional options on the right. As mentioned above, people usually prefer to focus on the top left corner of the page. However, we added an eye-catching logo to the top-right corner of the page so that when the page opens, the user's attention is drawn to the right, and the user will naturally find the data upload function area below the logo, which is the first step for using this application. Besides, we have placed the data submission button and the message box below the data preview window. Thus, when the visual components are successfully rendered, the user's attention is naturally directed to the summary statistics line chart at the bottom. Later, as the analysis is zoomed in to a specific period, the user's visual focus is gradually directed through a series of interactions, from the bottom to the node-edge network at the top-left and the adjacent statistics bar chart completing the entire process.

5. Visualization and Interaction

This section introduces the specific functions of each visual component of the functional area, together with the representation of measures and algorithmic implementation mentioned in the previous sections. Also, the linkages between the various visualization components and the interaction between the user and widgets are described to show an objective relation between data input and visual output.

5.1 Time-based summary

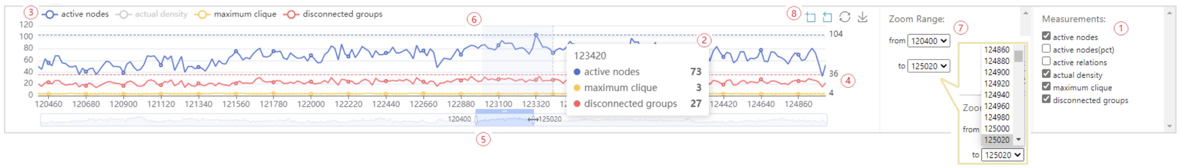

Figure 5.1: Visual component and widgets for time-based summary

An overview of the dataset is a necessary approach in the early stages of exploratory data analysis. Particularly to a time-series data of a social network, the time-based line chart could provide a clear picture of the development tendency of the network structure and the general behavior pattern of the elements in the network. Fig.5.1 shows the visual component and widgets of the functional area for a time-based summary. The user is allowed to select the metric of interests via checkboxes in the widget-1. Corresponding measures and algorithms are executed for each discrete network over time. The results are plotted simultaneously on a line chart so that the user could compare different metrics using a tooltip of widget-2 while hovering. Widget-3, the legend of the metrics, allows the user to mute and unmute the selected metrics in case some lines are flattened. In addition, the position of the global maximum is highlighted for each metric by a horizontal line, and the value is display at widget-4, which allows the user to retrieve the extreme value conveniently. Once users need to take a close look at a certain period, we offer three different solutions for zoom in and out, with their advantages. Draggable data zoom (widget-5), its left and right borders can be dragged separately to adjust the range of the selection, and horizontal sliding of the selected range is supported. Data brush (widget-6) works directly on the line graph, and although it lacks the precision of selection, it is the most convenient method in practice. Drop-down range select (widget-7) is the most accurate of the data boundary selection solutions and is used to display the specific values of the selected data range. Lastly, a toolbox (widget-8) is provided to reset the zooming operation and allow the user to export the line chart by image.

5.2 Sliding window selector

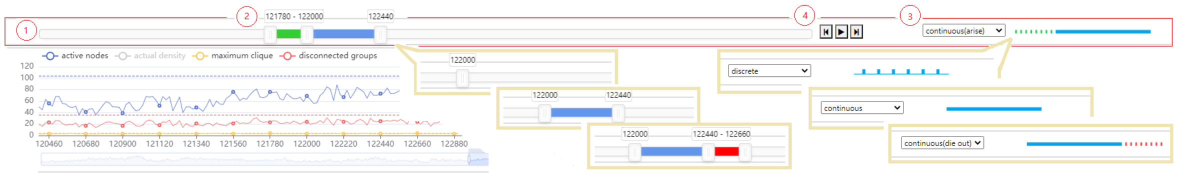

Figure 5.2: Visual component and widgets for sliding-window selector

The components and widgets for sliding window are managed as Fig.5.2. The implementation of the sliding window is adapted by a range slider (widget-1). Since the sliding window's timeline is based on the time range selected in the previous step from the line chart. The first technical challenge we conquered in this functional area is to ensure that the value range of the slider must match the value range of the x-axis of the line chart and that the scale of the slider must correspond to the scale of the x-axis of the line chart in the vertical direction. Secondly, we realize the dynamic update of the slider bar by listening to the events of the data zooming of the line chart so that the value range and scale ticks of the slider bar change in sync with the x-axis of the line chart.

In addition, we overwrite the source code of the range slider to make the component compatible with more than two handles (widget-2). Different modes of network operation could be select via a drop-down list in widget-3. Under 'discrete' mode, one handle works on selecting the discrete-time network over the timeline. Under basic "continuous" mode, the sliding window consists of two handles on the timeline. Therefore, all discrete-time networks which fall in the sliding window from T_{handle_1} to T_{handle_2} are merged into one continuous-time network. While in "continuous(arise)" mode, three handles are placed on the bar to form two adjacent sliding windows. The continuous-time network G as the object of study starts from T_{handle_2} until T_{handle_3} with blue solid line symbol, where an auxiliary continuous-time network(G^{-}) starts from T_{handle_1} until $T^{-}_{handle\_2}$ with green dash line symbol, therefore, the new raised entities and relationships could be obtained by $G - G^{-}$. Similar to the "continuous(die out)" mode, the object of study G from T_{handle_1} until T_{handle_2} and the auxiliary G^{+} from $T^{+}_{handle\_2}$ until T_{handle_3}, so that we have dying out elements by $G - G^{+}$.

Furthermore, we add a button component in widget-4 and manipulate the boundary of the sliding window by changing the position of the handles to simulate the animation effect. It allows the user to slide the window forward, backward, and autoplay with a simple click.

5.3 Node-edge network

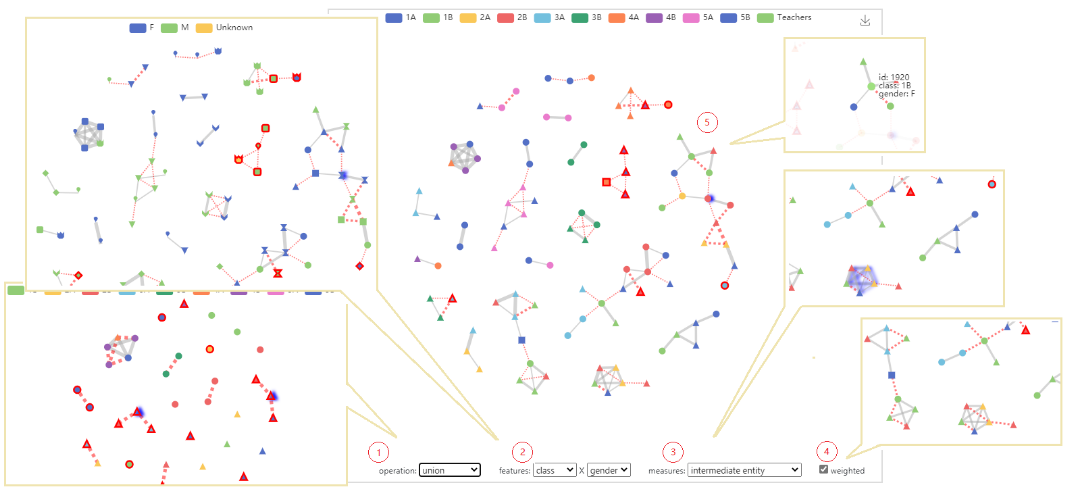

Figure 5.3: Visual component and widgets for node-edge network

The visual component of the node-edge network is also dynamically updated by listening to the events from the sliding window component. Besides the same functions on legend and image export, it displays the changes of the network structure and essential metrics of individual and population levels by rendering entities and relationships with different visual elements, such as shape, color, shadow, border, etc. Combining different features with various metrics allows the user to retrieve information from the displayed continuous-time network conveniently. The widget-1 of Fig.5.3 allows the user to switch the merging graph operation between union and intersection. The options in the drop-down list of widget-2 refer to all features of entities. The selected feature in the first drop-down list is categorized by different node colors in-network, which associates with the clickable legend. Different node shapes categorize the feature in the second list. We overload the shape drawing function of source code, that extend the total number of the shape from 9 to 15 by drawing new distinct shape such as tulip, bow tie, hourglass, etc. By analyzing categorized features, especially the interaction of two features, the user could get a clear picture of the similarities between those connected entities and the differences between those disconnected. The widget-3 allows users to select shadowing on either the entities with the highest score on node betweenness centrality, or the relationships with the highest score on edge betweenness centrality, or the maximum clique in the network. Therefore, users could quickly identify the bridge on spreading infection path or control the people in the cluster who may be exposed to a potentially high risk caused by cross-infection. In addition, the check box in widget-4 allows highlighting the long-existing relationships by transforming the network into weight. Last but not least, by hovering on the elements in the network graph, the information about the features entity and relationship is labeled, particularly the node, the degree, and all neighbors are highlighted.

5.4 Statistics of network

The stack bar chart in Fig.5.4 is adopted to present the distribution of active entities in the selected continuous-time network by categorized features. It complements the statistical aspects of the node-edge graph.

The feature in the first drop-down list in widget-1 refers to the category by bars, and the second drop-down refers to the category by color, which also associates with the legend. If there are so many categories of entities that the category names on the axis are compressed and cannot be fully rendered, we add a transpose function to the plot to choose to display the bars either vertically or horizontally. Same as others components, the image export function is also available.

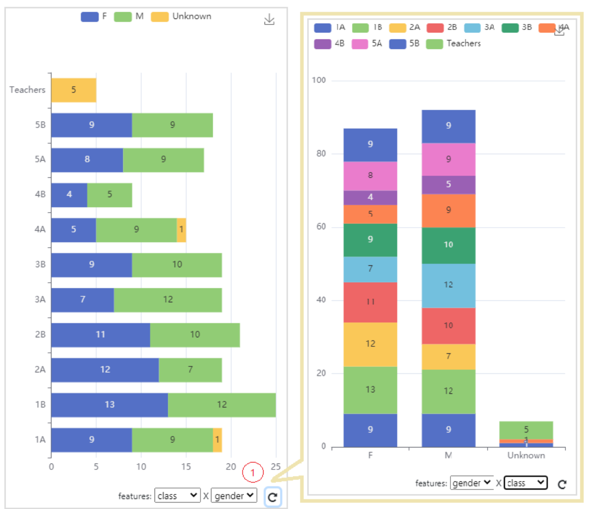

Figure 5.4: Visual component and widgets for statistics of network

6. Results

6.1 Results from use case

The use case testing is applied on the school activity dataset to verify the actual functionalities of the approach in practice.

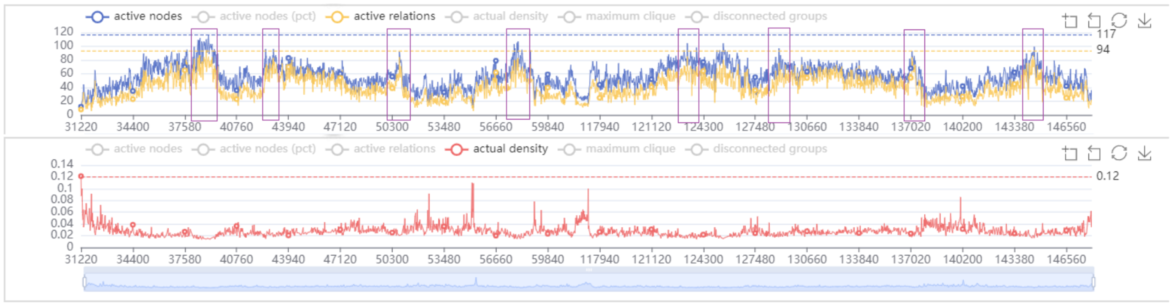

Figure 6.1: Patterns in time-based summary of school activity dataset

The first step is to observe the line chart of the time-based summary in Fig.(6.1). The pattern of active entities and active relationships showed eight peaks during the two days of activity. They were located around 10:30, 11:45, 14:00, and 16:00 each day, corresponding to the recess between classes. However, there is no clear pattern in the actual density of the network. The peak occurs when the active entities and relationships decline because the density is obtained by the proportion of the actual number of relationships to the potential relationships. Further, in a dynamic network where the number of entities varies over time, the number of potential relationships decreases polynomially as the number of entities decreases.

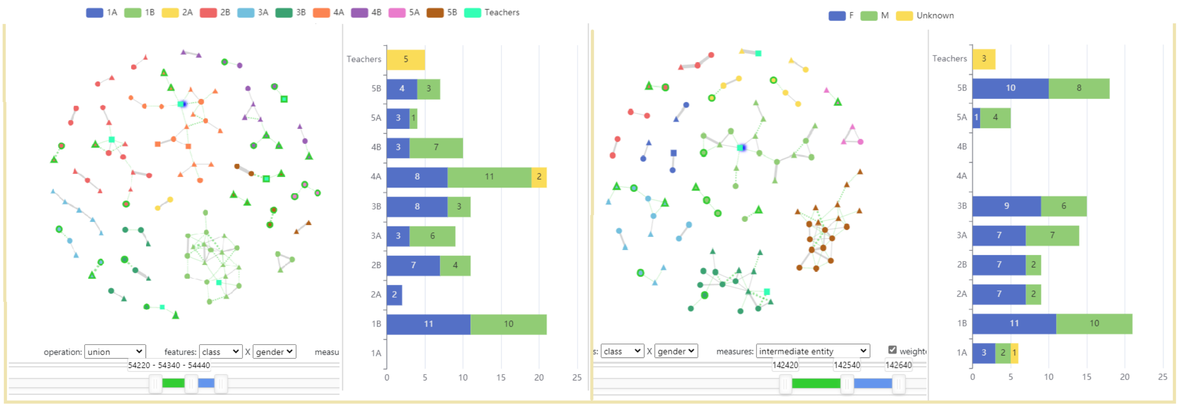

Figure 6.2: Node-edge graph for the continuous-time network with time span 2 minutes during lecture time. Left: sample from the morning lecture. Right: sample from the afternoon lecture

Next, the comparison of differences between the network structure at recess and during lecture time is proceeded by taking a close look at the node-edge graph through data zooming. We take a continuous-time network with a duration of two minutes on the timeline for further observation. Fig.6.2 shows sampled network structure during lecture time. There is no significant difference in the number of active students by gender from the stacked bar chart. The node-edge graph shows that the interaction of relationships between entities mainly occurs within the same class. Those disconnected sub-graphs of a certain size are composed of teachers in the center. In addition. Most of the relations are newly raised, and only a few relationships with high weight. It illustrates that the duration of a certain activity of an individual is short, and secondly, the interaction between two distinct individuals ends quickly. This implies a smaller probability of individuals exposed to infection. Moreover, teachers are more likely to be critical mediators of infectious disease transmission in the classroom than students.

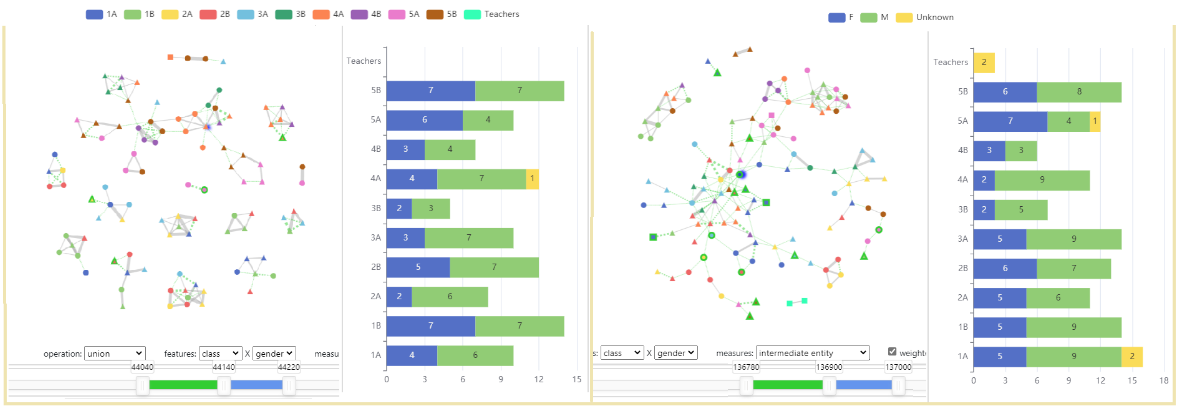

Figure 6.3: Node-edge graph for continuous-time network with time span 2 minutes at recess. Left: sample from lunch break. Right: sample from recess between two lectures

The left-hand side of Fig.6.3 shows the network structure during lunch break. The most obvious point is that the interaction between students is not limited to those from the same class. Such a pattern of cross-class interaction is that students prefer to stay with the student in the same grade or a similar grade. In addition, the proportion of newly risen active individuals and active relationships during lunch break is significantly less than during the lecture time. Besides, the network contains more high-weighted relationships. This implies that the interaction between students is more stable during the lunch break with a longer duration. The right-hand side of Fig.6.3 shows the network structure from recess between two-afternoon lectures. The teacher is no longer at the center of the network but instead tends to stray to the periphery, while interactions between students are more intense and shorter, possibly since students prefer to take a short break at the common area.

The results of our approach tell us that students are more likely to be exposed to infections during recess, especially during lunch break. Moreover, the spread of infectious diseases is more rapid among students in the same class, in the same grade, or a similar grade. Therefore, during the pandemic, the recommended measurement for schools to control infection could be: firstly, to limit non-essential contact and keep social distancing at recess. Secondly, when students are confirmed to be infected. It is considered to only close grades instead of close schools.

6.2 Discussion and limitation

During the above practical use, some minor drawbacks of the application were exposed.

- Some metrics, for example, the actual density, may not be informative for a dynamic network due to the entities and relationships vary over time.

- In the process of picking time using the sliding window, the control refreshes frequently and triggers merging operations on discrete networks, leading to unnecessary operations and application lag.
- Since the application interface hosts multiple visual components, the rendering of line graphs may be distorted when dealing with extremely large amounts of data.

All mentioned above will be our future work, and additional techniques may be necessary to deal with these challenges.

7. Conclusion

In this paper, we propose a web-based application for continuous-time network visualization. The proposed application is a dashboard-like visualization tool in which the surface displays data in a series of interactive visualization such as line charts, stacked bar charts, and node-edge graphs. Behind the scenes, it incorporates both data processing, social network metrics computation, and algorithms implementation to provide a general approach for increasing utility and universal usability of continuous-time network analysis.

Firstly, by reviewing the relevant literature on continuous-time network visualization, we summarize the advantages and disadvantages of existing visualization tools. Based on this, we employ a combined data structure of hash tables and adjacency lists and justify operational complexity, which is also the fundamentals of our application.

In this application, we also implement the sliding window by reconstituting the range slider's application component and using it to slice the discrete-time series data, and then through the corresponding graph merging algorithm, output a continuous-time network. In addition, we introduce a secondary sliding window adjacent to the observed sliding window on the timeline, thus enabling the highlighting of network structure changes over time by computing the difference in between. We associate all visual components through the sliding window. It provides the user a complete workflow in the logical order of data analysis on a continuous-time network from the population level to the individual level.

Furthermore, we chose some essential social network metrics to measure the data and render the visualization from the perspective of epidemiological studies. This allows our approach to function in specific areas of expertise. Through use case testing on school activity dataset during the H1N1 pandemic, it shows our approach can effectively display the patterns and trends of the overall structural changes of the social network within every single day of school activity, while also providing valid information for identifying individuals and subgroups that cause structural changes. In addition, the highlighting of network metrics can give valuable recommendations for epidemiological studies, especially for controlling and interrupting the spread of infectious diseases.

As social network analysis is widely used in various industry sectors, we have reasons to believe that by analyzing and studying user needs in different fields, our approach for continuous-time network visualization can be applied to more different research directions through further extensions and iterations in the near future.

References

Bender-Demoll, S. & Mcfarland, D. (2002). *Interaction, Time, and Motion: Animating Social Networks with SoNIA Introduction* (Tech. Rep.). Retrieved from `http://sonia.stanford.edu/exampleMovies/` 5, 8

Bloodgood, J. M., Hornsby, J. S., Rutherford, M. & McFarland, R. G. (2017). The role of network density and betweenness centrality in diffusing new venture legitimacy: an epidemiological approach. *International Entrepreneurship and Management Journal*. doi: 10.1007/s11365-016-0412-9 10

Bondy, J. A. & Murty, U. S. R. (1976). *Graph theory with applications.* New York: Elsevier. 11

Brandes, U., Raab, J. & Wagner, D. (2001). Exploratory Network Visualization : Simultaneous Display of Actor Status and Connections. *Journal of Social Structure*. 1

Bron, C. & Kerbosch, J. (1973). Finding all cliques of an undirected graph (algorithm 457). *Commun. ACM*, *16*(9), 575–576. 11

Cattuto, C., van den Broeck, W., Barrat, A., Colizza, V., Pinton, J. F. & Vespignani, A. (2010). Dynamics of person-to-person interactions from distributed RFID sensor networks. *PLoS ONE*. doi: 10.1371/journal.pone.0011596 6

Chiesi, A. (2001). Network analysis. In N. J. Smelser & P. B. Baltes (Eds.), *International encyclopedia of the social behavioral sciences* (p. 10499-10502). Oxford: Pergamon. Retrieved from `https://www.sciencedirect.com/science/article/pii/B008043076704211X` doi: https://doi.org/10.1016/B0-08-043076-7/04211-X 1

Everton, S. (2004, 01). A guide for the visually perplexed: Visually representing social networks.
1, 4

Fessenden, T. (2017). *Horizontal Attention Leans Left.* Retrieved 2021-06-09, from `https://www.nngroup.com/articles/horizontal-attention-leans-left/` 14

Flask. (2010). *Welcome to Flask — Flask Documentation (2.0.x).* Retrieved 2021-06-02, from `https://flask.palletsprojects.com/en/2.0.x/` 12

Freeman, L. C. (1977). A Set of Measures of Centrality Based on Betweenness. *Sociometry*. doi: 10.2307/3033543 10

Freeman, L. C. (2000). Visualizing Social Groups. In *Journal of social structure.* 1

Gemmetto, V., Barrat, A. & Cattuto, C. (2014). Mitigation of infectious disease at school: Targeted class closure vs school closure. *BMC Infectious Diseases*. doi: 10.1186/s12879-014-0695-9 6

Girvan, M. & Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, *99*(12), 7821–7826. Retrieved from `https://www.pnas.org/content/99/12/7821` doi: 10.1073/pnas.122653799 11

Hadlak, S., Schulz, H. J. & Schumann, H. (2011). In situ exploration of large dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*. doi: 10.1109/TVCG.2011.213 1, 3

Huang, K. E., Lipsitch, M., Shaman, J. & Goldstein, E. (2014). The US 2009 A(H1N1) influenza epidemic: Quantifying the impact of school openings on the reproductive number. *Epidemiology*. doi: 10.1097/EDE.0000000000000055 6

Joshi, V. (2018). *Seven Reasons Why A Website's Front-End And Back-End Should Be Kept Separate.* Retrieved 2021-06-08, from `https://www.forbes.com/sites/forbestechcouncil/2018/07/19/seven-reasons-why-a-websites-front-end-and-back-end-should-be-kept-separate/?sh=242613c04fca` 13

Liu, Q., Hu, Y., Shi, L., Mu, X., Zhang, Y. & Tang, J. (2015). EgoNetCloud: Event-based egocentric dynamic network visualization. In *2015 ieee conference on visual analytics science and technology, vast 2015 - proceedings.* doi: 10.1109/VAST.2015.7347632 1

Lloyd-Smith, J. O., Schreiber, S. J., Kopp, P. E. & Getz, W. M. (2005). Superspreading and the effect of individual variation on disease emergence. *Nature*. doi: 10.1038/nature04153 10

Longini, I. M., Koopman, J. S., Monto, A. S. & Fox, J. P. (1982). Estimating household and community transmission parameters for influenza. *American Journal of Epidemiology*. doi: 10.1093/oxfordjournals.aje.a113356 6

Moody, J., McFarland, D. & Bender-DeMoll, S. (2005). *Dynamic network visualization.* doi: 10.1086/421509 1, 3, 5, 8, 9

Moon, J. & Moser, L. (1965). On cliques in graphs. *Israel Journal of Mathematics*, *3*, 23-28. Retrieved from `http://dx.doi.org/10.1007/BF02760024` (10.1007/BF02760024) 11

Moreno, J. L. (1953). *Who shall survive? Foundations of sociometry, group psychotherapy and socio-drama.* 1

Mossong, J., Hens, N., Jit, M., Beutels, P., Auranen, K., Mikolajczyk, R., . . . Edmunds, W. J. (2008). Social contacts and mixing patterns relevant to the spread of infectious diseases. *PLoS Medicine*. doi: 10.1371/journal.pmed.0050074 6

Newman, M. E. (2002). Spread of epidemic disease on networks. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*. doi: 10.1103/PhysRevE.66.016128 10

Ognyanova, K. (2017). *Static and dynamic network visualization with R - Katya Ognyanova.* Retrieved 2021-05-22, from `https://kateto.net/network-visualization` 5

Pan, B., Hembrooke, H. A., Gay, G. K., Granka, L. A., Feusner, M. K. & Newman, J. K. (2004). The determinants of web page viewing behavior: An eye-tracking study. In *Eye tracking research and applications symposium (etra).* 14

Pang, X., Cao, Y., Lau, R. W. H. & Chan, A. B. (2016). Directing user attention via visual flow on web designs. *ACM Transactions on Graphics (TOG)*, *35*, 1 - 11. 14

Pernice, K. (2014). *How people read on the web: The eyetracking evidence.* Nielsen Norman Group. Retrieved from `https://books.google.nl/books?id=nQ2-oQEACAAJ` 14

Rajaraman, R. (2006). *Introduction to Dynamic Networks Models, Algorithms, and Analysis* (Tech. Rep.). Retrieved from `www.ccs.neu.edu/home/rraj/Talks/DynamicNetworks/DYNAMO/` 1

Schiller, B., Deusser, C., Castrillon, J. & Strufe, T. (2016). Compile- and run-time approaches for the selection of efficient data structures for dynamic graph analysis. *Applied Network Science*. doi: 10.1007/s41109-016-0011-2 7

Shi, L., Wang, C. & Wen, Z. (2011). Dynamic network visualization in 1.5D. In *Ieee pacific visualization symposium 2011, pacificvis 2011 - proceedings.* doi: 10.1109/PACIFICVIS.2011.5742388 1, 3, 4

Shi, L., Wang, C., Wen, Z., Qu, H., Lin, C. & Liao, Q. (2015). 1.5d egocentric dynamic network visualization. *IEEE Transactions on Visualization and Computer Graphics*, *21*(5), 624-637. doi: 10.1109/TVCG.2014.2383380 1

Silk, M. J., Croft, D. P., Delahay, R. J., Hodgson, D. J., Boots, M., Weber, N. & McDONALD, R. A. (2017). *Using social network measures in wildlife disease ecology, epidemiology, and management.* doi: 10.1093/biosci/biw175 10

Smieszek, T. (2009). A mechanistic model of infection: Why duration and intensity of contacts should be included in models of disease spread. *Theoretical Biology and Medical Modelling*. doi: 10.1186/1742-4682-6-25 10

Snijders, T. (2005). Models for longitudinal network data. *Models and Methods in Social Network Analysis*. 1

Weber, N., Carter, S. P., Dall, S. R., Delahay, R. J., McDonald, J. L., Bearhop, S. & McDonald, R. A. (2013). *Badger social networks correlate with tuberculosis infection.* doi: 10.1016/j.cub.2013.09.011 10

Willis, G. A., Preen, D. B., Richmond, P. C., Jacoby, P., Effler, P. V., Smith, D. W., . . . Blyth, C. C. (2019). The impact of influenza infection on young children, their family and the health care system. *Influenza and other Respiratory Viruses*. doi: 10.1111/irv.12604 6