

Department: Head
Editor: Name, xxxx@email

DiffSeer: Difference-based Dynamic Weighted Graph Visualization

Xiaolin Wen

Sichuan University, Chengdu, China and Singapore Management University, Singapore

Yong Wang

Singapore Management University, Singapore

Meixuan Wu, Fengjie Wang

Sichuan University, Chengdu, China

Xuanwu Yue

Sinovation Ventures AI Institute, Beijing, China and Baiont Technology, Nanjing, China

Qiaomu Shen, Yuxin Ma

Southern University of Science and Technology, Shenzhen, China

Min Zhu

Sichuan University, Chengdu, China

Abstract—Existing dynamic weighted graph visualization approaches rely on users’ mental comparison to perceive temporal evolution of dynamic weighted graphs, hindering users from effectively analyzing changes across multiple timeslices. We propose *DiffSeer*, a novel approach for dynamic weighted graph visualization by explicitly visualizing the *differences* of graph structures (e.g., edge weight differences) between adjacent timeslices. Specifically, we present a novel nested matrix design that overviews the graph structure differences over a time period as well as shows graph structure details in the timeslices of user interest. By collectively considering the overall temporal evolution and structure details in each timeslice, an optimization-based node reordering strategy is developed to group nodes with similar evolution patterns and highlight interesting graph structure details in each timeslice. We conducted two case studies on real-world graph datasets and in-depth interviews with 12 target users to evaluate *DiffSeer*. The results demonstrate its effectiveness in visualizing dynamic weighted graphs.

arXiv:2302.07609v1 [cs.HC] 15 Feb 2023

■ **DYNAMIC WEIGHTED GRAPHS** model the temporal evolution of detailed relationships between entities in various applications such as social networks, financial networks, or communication networks. To analyze such dynamic weighted graphs, a large number of dynamic graph visualization techniques have been proposed, where the essential research question is to investigate how to visualize *the temporal changes* of dynamic graph structures effectively [1], [20]. Most existing dynamic graph visualization approaches focus on displaying graph structures (i.e., nodes and edges) along the time via either animated diagrams or a series of static charts (e.g., small multiples) [9]. It is non-trivial to compare the differences between weighted graphs [2]. To explore the temporal evolution patterns of dynamic weighted graphs, users need to *mentally compare* the differences between multiple adjacent timeslices simultaneously, which is more challenging.

Different from prior studies for dynamic graph visualization, we aim to achieve effective dynamic weighted graph visualization across a period of time from a new perspective: *explicitly visualizing the differences between adjacent timeslices*. Such a new perspective has clear advantages. As shown in Figure 1a, it's hard to quickly identify how the graph structures (e.g., edge weights) are exactly changing between two adjacent timeslices through mental comparison. When explicitly encoding the differences of edge weight by the width and using red and blue to represent the positive and negative changes (Figure 1b), the temporal changes of dynamic weighted graphs can be seen directly and do not rely on the mental comparison. Figure 1c shows another dynamic weighted graph that is different from Figure 1a, but they have the same difference sequence (Figure 1b).

However, it is non-trivial to leverage differences for dynamic graph visualizations. First, graph differences can refer to different aspects of graphs such as the changes of edge weights. How to visualize such graph differences in a universal way needs further exploration. Second, the temporal evolution of dynamic weighted graphs relies on understanding multiple graph differences of continuous adjacent timeslices. It remains unclear regarding how to visualize such multiple graph

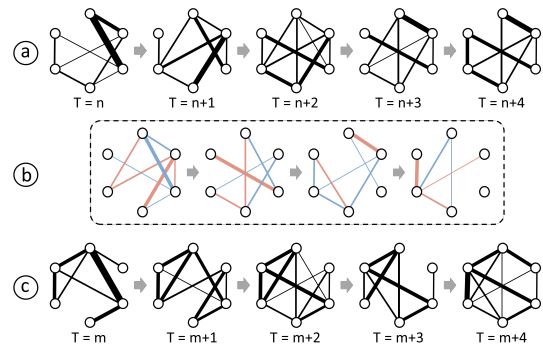


Figure 1. Illustration of the importance of differences in dynamic weighted graph visualization. (a) and (c) show two different dynamic weighted graphs. (b) shows the graph differences between adjacent timeslices of both (a) and (c), where the edge width encodes the edge weight changes and the color represents the trend of changes (i.e., red for an increase and blue for a decrease). (b) provides a direct perception of temporal changes in dynamic weighted graphs.

differences over time and highlight the temporal patterns like reoccurring and outliers [19]. Third, despite the importance of graph differences, graph structures themselves are also useful for a comprehensive interpretation of dynamic weighted graphs. It is challenging to inform users of both graph differences and original dynamic weighted graph structures effectively.

In this work, we propose *DiffSeer*, a novel difference-based approach for dynamic weighted graph visualization (Figure 2). It can effectively inform users of the overall temporal evolution of dynamic weighted graphs and show the graph structure details in individual timeslices. Specifically, we present a novel *nested matrix design* that overviews the graph differences over a period of time and enables interactive inspection of graph details on demand. Given the essential information of dynamic weighted graphs is edge weights [8], we leverage edge weight differences to represent graph differences. A new *node re-ordering strategy* is proposed to group nodes in the nested matrix design, which can manifest the overall temporal evolution patterns and interesting graph structure details of individual timeslices in a balanced manner. A *difference mask* is integrated into the nested matrix design

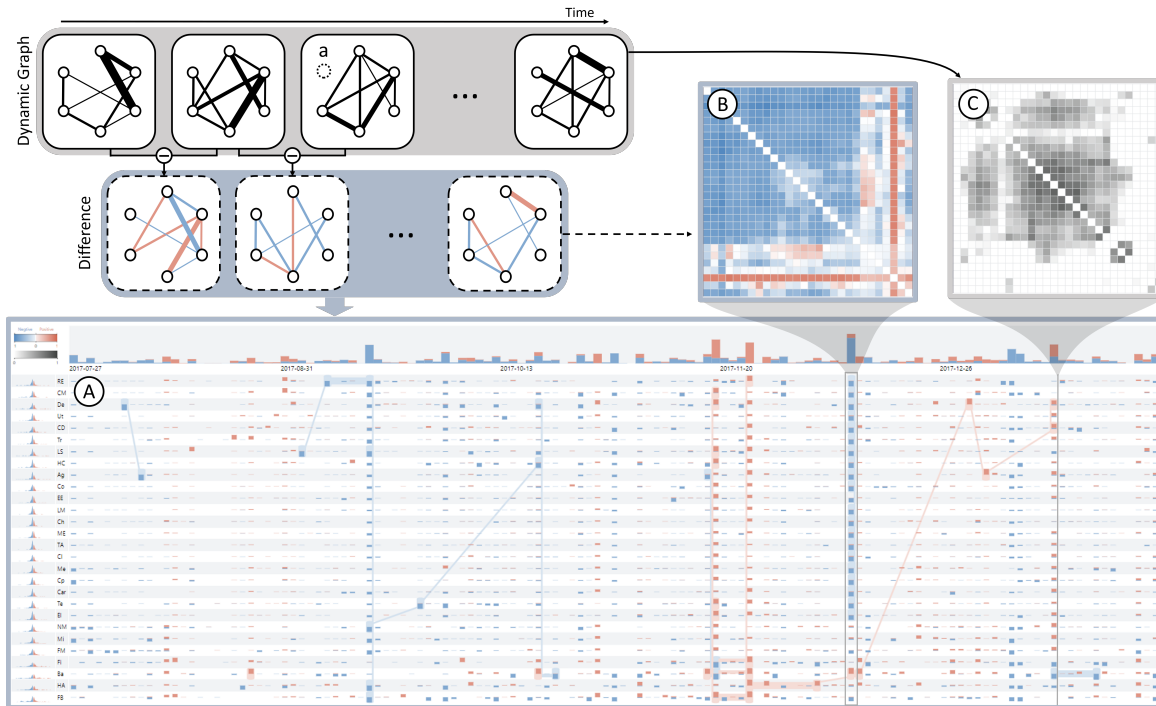


Figure 2. Overview of *DiffSeer*: We focus on explicitly visualizing the differences between adjacent timeslices to support the analysis of the dynamic weighted graph evolution over a long time. Specifically, we proposed a *nested matrix design*, including (A) an *overview matrix* to provide a visual summary of differences and two types (B, C) of *detail matrices* to enable interactive inspection of graph details on demand. An optimization-based *node reordering strategy* is incorporated in the nested matrix design to group together nodes with similar evolution patterns and highlight interesting graph structure details in each timeslice.

to enhance the perception of interesting temporal evolution patterns. We conduct two case studies on real-world dynamic graph datasets and in-depth interviews with 12 expert users. The results demonstrate the effectiveness and usability of *DiffSeer* in informing viewers of the temporal evolution of dynamic weighted graphs.

In summary, our main contributions include:

- We propose *DiffSeer*, a novel difference-based dynamic weighted graph visualization approach, for exploring the dynamic graph evolution interactively, where a novel nested matrix design is presented to show the temporal evolution of dynamic graphs with multiple levels of details.
- We conduct two case studies on real-world datasets and user interviews with 12 target users to demonstrate the effectiveness and usability of *DiffSeer*.

Related Work

Dynamic graph visualization has been extensively studied in the past decades. The major dynamic graph visualization methods include animation-based and timeline-based approaches [9]. Animation-based approaches show the temporal evolution through animated transitions [14]. Timeline-based approaches visualize dynamic graphs through a time-to-space mapping, where the graph structure of each timeslice is shown along a timeline [6], [8]. Most of these approaches focus on displaying the graph structures in each timeslice directly. To identify the temporal evolution patterns of dynamic graphs, users need to mentally compare the changes or differences between adjacent timeslices in the animated graphs or between a sequence of static graphs [4]. For dynamic weighted graphs, such a process relies on users' mental memory due to the complex weight changes, hindering users from effectively and quickly analyzing the temporal

changes.

A few prior studies have also investigated highlighting the removed or inserted edges and nodes between two adjacent timeslices in small multiples [3], [13] and animated transitions [7], [11]. Archambault et al. [5] further performed a user study on such methods and showed that highlighting changing edges is helpful. However, these studies focus on using color to indicate whether an edge occurred or not and are unable to reveal change details in dynamic weighted graphs such as edge weight [8]. For dynamic weighted graph, users need to understand both the edge weight differences and the original graph structures. Previous studies have never explored leveraging detailed edge weight differences to display the evolution of dynamic weighted graphs.

Different from prior studies, *DiffSeer* is a novel difference-based visualization approach for dynamic weighted graphs. It explicitly visualizes the edge weight differences over time, and enables systematic exploration of the temporal evolution of dynamic weighted graphs with multiple levels of details.

Background

In this paper, we focus on the undirected dynamic weighted graph and give concrete definitions as follows. A *dynamic weighted graph* Γ can be regarded as a sequence of graph snapshots G in each timeslice:

$$\Gamma = \{G_1, G_2, \dots, G_T\}, \quad (1)$$

where each graph snapshot $G_i = (V_i, E_i)$ has a different node set $V_i = \{v_1, v_2, \dots, v_N\}$ and a different edge set $E_i = \{e_1, e_2, \dots, e_M\}$ in each timeslice. Each edge e_m is defined as follows:

$$e_m = (u, w, v) \in E_i \subseteq V_i \times R^+ \times V_i, \quad (2)$$

where $u, v \in V_i$ are two nodes linked by e_m , w denotes the weights of edges in the graph.

We define the node set V as all the nodes that appear at least once throughout the whole time range, and define the *graph difference* between two adjacent timeslices of a dynamic weighted graph as follows:

$$Diff_i = (V, D_i), \quad (3)$$

where D_i is the set of edges with weight changes in the i -th timeslice. $Diff_i$ is intrinsically a

graph as well. Specifically, D_i is defined as:

$$D_i = \{d_1, d_2, \dots, d_H\}, 2 \leq i \leq T, \quad (4)$$

where H is the total number of edges with a weight change from G_{i-1} to G_i , and each changed edge d_k is:

$$d_k = (u, w'_k, v), 1 \leq k \leq H, \quad (5)$$

where w'_k is the edge weight change of the k -th edge between G_{i-1} and G_i and can be positive or negative. A positive edge weight change indicates an increase of edge weight, while a negative one indicates a decrease of edge weight.

Like existing dynamic weighted graph visualizations [20], [8], we focus on the changes of edge weight, one of the important features of dynamic weighted graphs. It is possible that some nodes may also appear or disappear in a dynamic weighted graph at some timeslices, which can be indicated by the appearance or disappearance of their associated edges. The dotted node in Figure 2a shows an example of node disappearance.

DiffSeer

We propose *DiffSeer*, a novel difference-based approach for dynamic weighted graph visualization. Its core component is a nested matrix design (Figure 3B) to provide both an overview and fine-grained details of edge weight differences. A difference mask (Figure 3E) can be interactively enabled to emphasize the significant changes, and stacked bar charts (Figure 3b₂) and area charts (Figure 3b₃) to overview the edge weight changes at each timeslice and associated edge weight change distribution of individual node respectively. Besides the nested matrix design, we also present a timeline view (Figure 3A) to enable a temporal summary of the original dynamic graph. It is achieved by projecting the graph structures at each timeslice into one dimension and the offset on the vertical axis can be a rough indication of the changing intensity [19], providing a quick insight into the entire period. A prototype system of *DiffSeer* is available at <https://diffseer.github.io/>.

Nested Matrix Design

This section introduces the visual design and the node reordering strategy of the nested matrix design in detail.

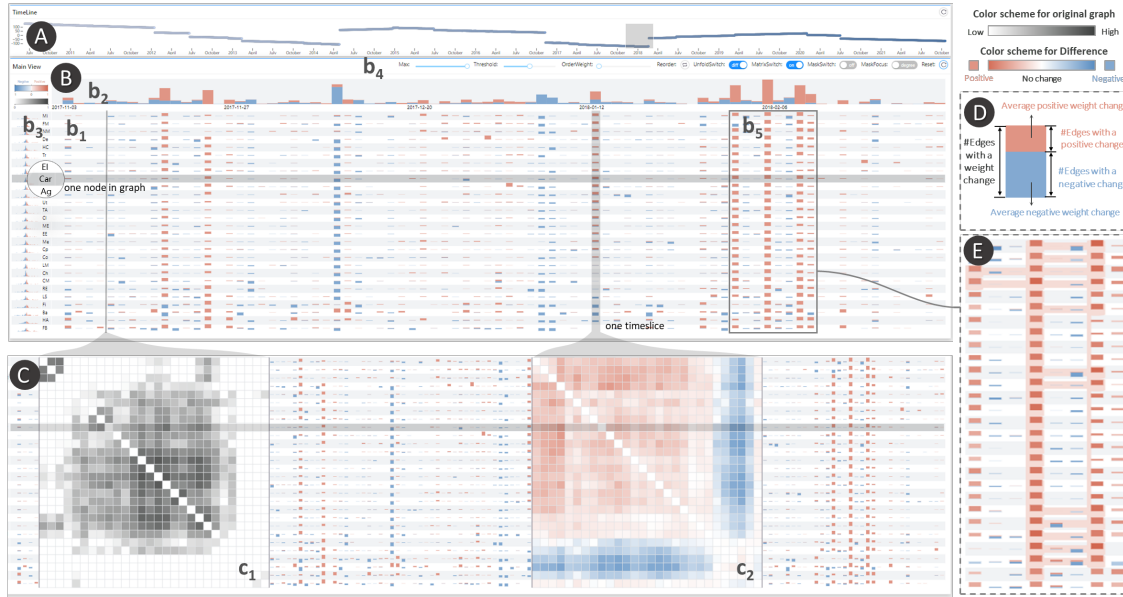


Figure 3. The *DiffSeer* interface consists of (A) a timeline view and (B) a nested matrix design including (b_1) the nested overview and detail matrices, (b_2) a stacked bar chart, (b_3) some area charts representing the overview of changes in nodes weights, and (b_4) a toolbar to provide some necessary interactions. The rows and columns in the nested matrix represent nodes and timeslices respectively. (C) shows the nested matrix when an original detail matrix (c_1) and a difference detail matrix (c_2) are unfolded in the overview matrix. (D) is the explanation of each cell in the overview matrix. (E) shows the difference mask attached to (b_5) in the overview matrix.

Nested overview and detail matrices. The nested matrix design has two core parts: an *overview matrix* (Figure 3 b_1) shows the graph differences between adjacent timeslices over a period of time to inform users of the temporal evolution of dynamic graphs, and *detail matrices* (Figure 3 c_1, c_2) display the details of graph difference or original graph structure at individual timeslices when a user double-clicks a timeslice of the *overview matrix*.

Overview matrix. As shown in Equation 3, graph difference can also be regarded as a type of graph. To effectively visualize the overall graph differences along time within the limited space, we aggregate edge weight differences into their connected nodes in the overview matrix, which is an effective way of aggregation for graph visualization [8]. The rows of the overview matrix (Figure 3 b_1) represent graph nodes, the columns correspond to different timeslices, and each cell intuitively shows the weight changes of all edges connected to one node in the corresponding timeslice. For each cell (Figure 3D), we use a stacked-bar glyph to encode the number

of edges in the *graph difference*, i.e., the number of edges with a weight change between two adjacent timeslices in the original dynamic weighted graph. The height of the upper or bottom bars of a glyph encodes the number of edges with a positive or negative weight change, respectively. The color scheme of bars refers to average weight changes of all the edges connected to a node in the *graph difference*, and we use a diverging red-to-blue color scheme to encode it.

Detail matrix. To check the details of the differences or the context in which the differences occurred, we allow users to interactively unfold the detail matrices of individual timeslices within the overview matrix by double-clicking the corresponding timeslices, as shown in Figure 3C. We use the matrix layout to visualize graph details because the detail matrices can share the same node order with the overview matrix to help users maintain analysis continuity. The detail matrix has two types: the *difference detail matrix* (Figure 3 c_2) and the *original graph detail matrix* (Figure 3 c_1). The difference detail matrix displays the specific difference between adjacent times-

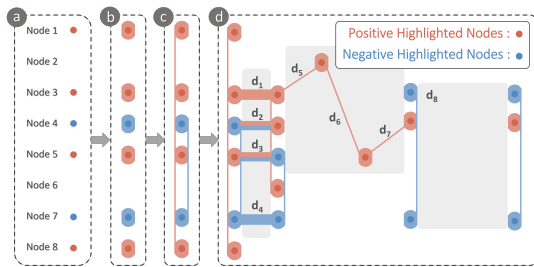


Figure 4. The drawing procedure of *difference mask*: (a) select nodes with significant changes, (b) highlight nodes, (c) connect highlighted nodes in one timeslice, (d) connect highlighted nodes cross timeslices.

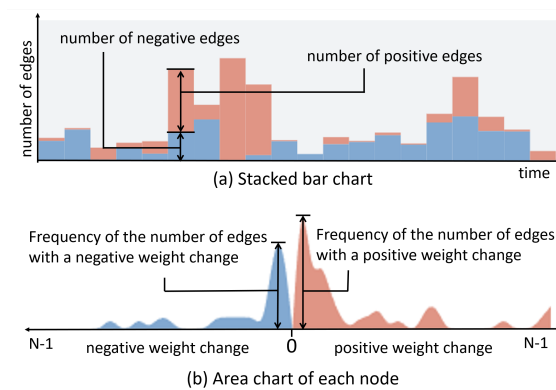


Figure 5. Visual encoding of stacked bar chart (a) and area chart (b) of each node. N is the number of nodes in the graph.

lices, where the rows and columns both represent nodes, and each cell represents a weight change of an edge. Similar to the overview matrix, we also use a diverging red-to-blue diverging color scheme here. The original graph detail matrix visualizes the graph structure in an individual timeslice, allowing users to inspect the exact graph structure details. A sequential gray-scale color scheme is used to encode the original edge weights.

Difference mask. To further facilitate an easy exploration of dynamic weighted graphs, we propose a difference mask (Figure 3E) to emphasize the temporal patterns of graph differences (e.g., outliers or repeated changes [19]). Specifically, we highlight the nodes with a significant edge weight change above a user-defined threshold by using red and blue rounded rectangles (Figure 4b). Here a significant edge weight change refers to either the average weight changes of

edges connected to a node or the total number of edges with a weight change. Then, we use paths to connect nodes with similar changes within and between timeslices, respectively. For one timeslice, we use vertical paths to join the highlighted nodes of the same color in a column, red for positive and blue for negative (Figure 4c), indicating that these nodes have similar significant changes. To emphasize the temporal patterns, paths are also drawn between the columns with at least one highlighted node. Some of the paths are horizontal connecting the same highlighted nodes of two columns to denote that similar changes appear repeatedly, of which color depends on whether the two nodes are positive or negative (Figure 4 d_{1-4}). In addition, when the number of consecutive columns with no highlighted nodes exceeds a predefined threshold, no path is drawn between them because this may imply a stable period (Figure 4 d_8).

Stacked bar chart and area chart. To provide the overall context, we add a stacked bar chart and some area charts to the nested matrix design. The stacked bar chart (Figure 5a) presents the distribution of the changed edges' number over time, which shares the same timeline as the overview matrix, and the height of the red part encodes the number of the positive edges while the blue part encodes the negative edges. The area charts (Figure 5b) present the distribution of the changed edges' number of each node during the selected period, which can imply which nodes always have many edges with a weight change.

Node Ordering Strategy

The node order of a matrix-based design can directly determine the visualization effect [17]. In nested matrix design, overview matrix and detail matrices share the same node order, so a proper node order is of vital importance. Figure 6a shows the overview matrix and detail matrices in an alphabetical node order. We can discover some patterns in the overview matrix, such as the timeslices with significant changes, but it is hard to get deep insights like some nodes with similar patterns. As for detail matrices, it is hard to figure out any interesting patterns beyond the value of each cell, such as clusters and subgroups.

To this end, we propose a node reordering strategy, which can simultaneously provide a sin-

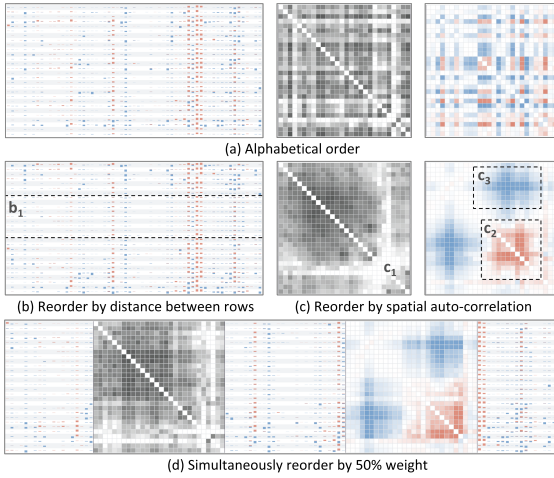


Figure 6. The figures showcase the nested matrix design with different node orders to show the effectiveness of our node reordering strategy. (a) order by alphabetical. (b) order by the distance between rows, mainly designed for the overview Matrix. (c) order by spatial auto-correlation, mainly designed for the detail matrix. (d) order by considering both overview matrix and detail matrix equally.

gle order that works well for both overview matrix and unfolded detail matrices. Users can adjust the reordering strategy’s priority for specific analysis needs. Figure 6b,6c shows the overview matrix and detail matrices after node reordering, respectively. Figure 6d shows the nested matrix after node reordering by treating the detail matrix and the overview matrix equally. Compared to Figure 6a, each of the ordered matrices can provide some additional insights, such as some nodes with similar changing patterns (Figure 6b₁), two nodes that have little relationship to other nodes (Figure 6c₁), and a group of nodes that have stronger connections within them (Figure 6c₂) but weaker connections with other nodes (Figure 6c₃). Further, the node reordering by equally considering both detail matrix and overview matrix can also preserve their own visual patterns as much as possible (Figure 6d).

Most existing methods reordered the matrix by calculating the inner-rows distance of the matrix and then solving it as a traveling salesman problem (TSP) [10]. Our strategy follows a similar framework, but the difference is that we need to find an appropriate order for different

types of matrices simultaneously. Specifically, we augmented the reordering method using Moran’s I [18], which is designed for unweighted and undirected dynamic graphs, to work for weighted dynamic graphs. Maximizing Moran’s I can be translated into minimizing the similarity I_s between two adjacent rows:

$$I = \sum_{a=1}^{n-1} I_s(M, \rho(a), \rho(a+1)), \quad (6)$$

where $\rho(a)$ denotes the a -th row of matrix M (n rows) in any node order and $I_s \in \left[-\frac{1}{n-1}, \frac{1}{n-1}\right]$. We normalize the distance between two rows as D_{detail} . For the overview matrix, since the spatial auto-correlation index can not work in the condition that the rows and columns are not symmetric, we only consider the distance between the two rows based on similarity. The inner-row distances are calculated with Manhattan distance as $D_{overview}$.

After getting the distances between rows of different types of matrices, the final node order should be calculated based on the user-defined weight to satisfy the user’s preference. We normalize the distance matrices of both the detail matrices and the overview matrix to the same scale and then set an adjustable weight parameter $\alpha \in [0, 1]$ to capture the user’s preference, and the final inter-row distance D is defined as:

$$D = \alpha D_{overview} + (1 - \alpha) D_{detail}, \quad (7)$$

which is used as the input of the leaf order algorithm [12]. It can calculate the final node order by constructing a hierarchical clustering on node rows according to the distance matrix.

When α is adjusted to 1, the reordering algorithm only focuses on the overview matrix (Figure 6b). On the contrary, when α is 0, nodes are reordered only based on Detail Matrices (Figure 6c). When α is set to a value between 0 and 1, such as 0.5, an overall suitable order may be obtained as shown in Figure 6d.

Interactions

DiffSeer enables rich interactions to allow users to smoothly analyze and explore dynamic weighted graphs. Users can brush the time range of their interest in the timeline view and explore the details in the nested matrix view. *DiffSeer* provides users with the flexibility of configuring the visual design, such as setting the reordering

weight and the difference mask threshold. More details of interactions can be found in the prototype system.

Case Study

We conducted two case studies to demonstrate the effectiveness of *DiffSeer* on real-world dynamic weighted graph datasets, which are collected from the domains of finance and social network. Two expert users (U1, U2) were involved in the case studies, and they have attended our user interviews and learned how to use *DiffSeer*. They were asked to use *DiffSeer* to explore dynamic weighted graphs. Their exploration procedures and the corresponding findings were recorded. We used two dynamic weighted graphs whose details are as follows:

Sector Index Correlation Network (SICN). We collected time-series records of 28 sector indices in the Chinese stock market¹. It consists of the stock trading records from June 3, 2010 to October 18, 2021 (2,761 timeslices), which covers the time period of the COVID-19 outbreak. We calculated the Pearson correlation between any two indices each day to model the association between different indices [15]. The sector indices are regarded as the nodes, and the correlations between each pair of sector indices are regarded as the edge weight.

Rugby Team Tweet Network (RTTN). This dataset contains more than 3,000 tweets between 12 teams in the Guinness Pro12 competition from 2014 to 2015 [16]. Each team is regarded as a node, and the edge weight denotes the number of tweets between two Rugby teams. There are 12 nodes and 329 timeslices in total.

Case 1: Impact of the COVID-19 on China Stock Market

U1 is a financial expert, and he was quite interested in the impact of COVID-19 on the correlation between different sector indices in China stock markets. So he brushed the period after the COVID-19 outbreak (i.e., from January 17, 2020, to March 13, 2020), as shown in Figure 7. From the overview matrix, U1 gained a quick understanding of the overall temporal evolution of the sector correlation network across

¹<https://www.joinquant.com>

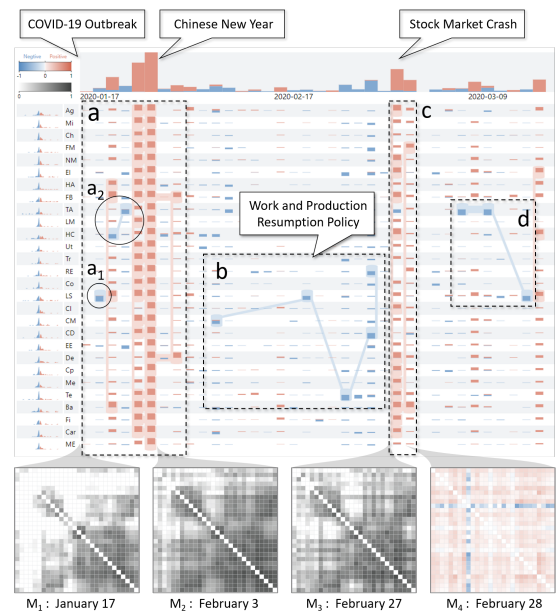


Figure 7. Overview matrix of the sector index correlation network dataset after the COVID-19 outbreak. (a) and (c) show a significant increase in the sector correlations. (b) and (d) show that several sectors have weakened their connection with others.

35 trading days. Specifically, he quickly noticed two periods with a significant increase in the sector correlations: the Chinese New Year period of 2020 and around early March, as indicated by the dense red bars in Figure 7a and Figure 7c.

Chinese New Year of 2020. With *DiffSeer*, U1 easily noticed that there were significant increases of edge weights for many nodes before Chinese New Year (Figure 7a), indicating a global correlation increase among different sectors. At the beginning of the outbreak of COVID-19, it was observed that Node *LS* (Leisure Service) first experienced a significant negative change (Figure 7a₁), followed by the blue difference mask between Node *HC* (Health Care) and Node *TA* (Textile & Apparel) (Figure 7a₂). It makes sense because Leisure Service (*LS*), a crowd-intensive sector, was first affected by the COVID-19 pandemic, and (*HC*) and (*TA*) declined in correlation with other sectors due to the shortage of treatment, masks, and other protective resources. U1 unfolded two original graph detail matrices to check the context at the beginning and end of this period. He found several clusters before the outbreak (Figure 7M₁), but almost all the

nodes were connected after the Chinese New Year (Figure 7M₂), indicating that these sectors were responding in a similar manner to the outbreak.

Work and production resumption. U1 further identified several nodes connected by the blue difference mask in Figure 7b. Except for these nodes, the edge weights of all the other sector nodes did not change very much during this period. According to U1, the reason for this pattern was a succession of national policies aimed at these changed sectors, i.e., these nodes all had similar changing patterns after the introduction of the policy to resume work and production. Observing the detail matrix after the resumption (Figure 7M₃), he found that the color of these rows was lighter than those in M₂ and confirmed that the work and production resumption did make some sectors less relevant to others.

Stock market crash. Next, U1 wanted to analyze the significant correlation increase in Figure 7c. He unfolded the difference detail matrix (Figure 7M₄) by double-clicking the corresponding timeslice. U1 found only an apparent blue column (row) of Node TA appearing in a wide range of red and noticed a negative difference mask on Node TA and Node LS after that day (Figure 7d). U1 mentioned that the global increase of edge weight may result from the stock market crash, and the interesting pattern on Node TA and Node LS is owing to the policy support that kept these sectors isolated from the negative influence of the whole market.

By using *DiffSeer*, U1 concluded that the COVID-19 pandemic has had a great impact on the industry association network, with significant fluctuations in closely related sectors. U1 said that gaining these insights would have taken much longer without using *DiffSeer*.

Case 2: Tweet Interactions between Rugby Teams

U2 has a background in social network analysis and is curious about the evolution of the Rugby team tweet network. With *DiffSeer*, U2 found interesting Tweet interaction patterns between Rugby teams from April 2015 to June 2015 (Figure 8).

Similar highlighted shapes. In the overview matrix, U2 found that some similar highlighted shapes were always separated by several columns

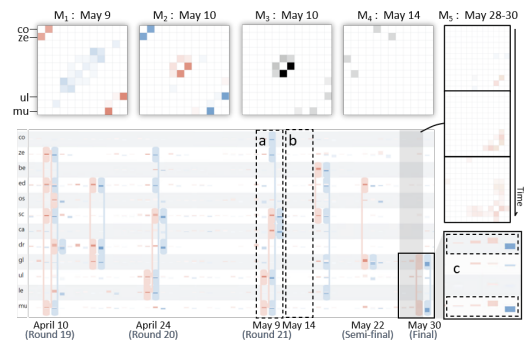


Figure 8. Dynamic evolution of the rugby team tweet network dataset from April to June. (a) and (b) show some significant changes occurred during the game, but few changes occurred after the game. (c) shows the changed edges' number of two nodes increased continuously before the finals.

with almost no changes, where positive edge weights (red rectangles) often appear before negative edge weights (blue rectangles), as shown in Figure 8. It indicates recursive repeats of abrupt increase followed by abrupt decrease of edge weights. To identify the reason behind it, U2 checked the schedule of the Rugby competitions for these teams². He found that these patterns are extremely relevant to the rounds of the competitions, i.e., there is a significant increase of the tweets between different Rugby teams on the start date of each round of competition, which would cease after the competition. To further check the interaction details, he unfolded two difference detail matrices in one of the highlighted shapes (Figure 8a) and saw that two pairs of nodes were dark red in Figure 8M₁ but dark blue in Figure 8M₂. It turned out that they were exactly the four teams participated in the two matches on that day, indicating the abrupt increase or decrease of interaction mainly come from the teams involved in the competition. Then, he compared the original graph detail matrices of a game day (Figure 8M₃) and a day without a competition (Figure 8M₄). He confirmed that the stable period (e.g., Figure 8b) was due to almost no tweets on days without games.

A pre-game hype. In Figure 8c, U2 found that the red bars of the two nodes kept increasing from May 27 to May 30, indicating an increas-

²https://en.wikipedia.org/wiki/2014-15_Pro12

ing interaction between the two teams and other teams. but they suddenly switched to blue bars on June 1st. By reordering nodes and comparing difference detail matrices, U2 found an obvious spread pattern of the two nodes, as shown in Figure 8M₅, which shows that more and more teams started to have tweet interactions with these two teams. According to the schedule, May 30 was exactly the final round date between the two teams, so U2 inferred that this pattern probably results from the pre-game hype, which is also why the connections between teams decreased rapidly after the finals.

Overall, with *DiffSeer*, U2 can figure out the deep correlation between the competition schedule and the evolution of the tweet interactions between Rugby teams.

User Interview

We conducted semi-structured interviews with 12 expert users who work on network data analysis and visualization to evaluate the effectiveness and usability of *DiffSeer*.

Participants and Apparatus

We invited 12 target users (U1-U12) to participate in the interviews. U1 and U2 are researchers from financial and internet companies, and U3-U12 are students from several universities (two doctors and eight masters). The participants (eight males and four females, aged 25 to 32, with normal vision) are engaged in network data analysis or visualization. Four (U1-U4) have more than five years of research experience, and others have at least one year. Due to the COVID-19 pandemic, our interviews with 7 participants were conducted online via Zoom, where they used their own computers to access *DiffSeer* deployed on a Cloud server. The interviews with the remaining participants were conducted offline on a desktop with a 23.8-inch 1920 x 1080 monitor.

Datasets and Tasks

The SICN and RTTN datasets were used in our user interviews. A small period of the RTTN dataset was used for the tutorial, while the other period of the RTTN dataset and the whole SICN dataset was used for the exploration by participants. During the interviews, the participants were

asked to complete the following five tasks to fully explore the dataset with *DiffSeer*.

T1. Find the changes occurring over time.

T2. Describe the type of changes.

T3. Find repeated changes in this period.

T4. Find the nodes with similar change patterns over time.

T5. Find the nodes with similar patterns in one timeslice.

T1 and T2 are designed to test whether the nested matrix design can accurately represent differences and help participants quickly identify graph structures that have changed over time. T3 is used to guide participants to discover some temporal patterns of differences with *DiffSeer* since the repetition of changes is one of the major temporal patterns [19]. T4 and T5 aim to evaluate the usefulness of our node reordering strategy. Since these tasks are subjective for users, we only use them to provide guidance but do not discuss the accuracy and time consumption.

Procedure

During the interview, we first introduced *DiffSeer* to the participants. Then, we went through an example usage scenario on the RTTN dataset to show how to use *DiffSeer* to explore a dynamic weighted graph. Then, the participants were allowed to explore the RTTN dataset freely to make themselves familiar with *DiffSeer*. The tutorial above lasted about 20 minutes. After that, they were invited to use *DiffSeer* to analyze either the RTTN dataset of another time period or the SICN dataset and finish the above tasks. Participants could freely explore the dataset until they felt that the tasks had been well finished. Their comments and suggestions were recorded. After that, we further invited them to finish a post-study questionnaire, including the set of questions in Table 1. Nine closed-ended questions (Q1-Q9) were designed to evaluate the usability (Q1-Q4) and effectiveness (Q5-Q9) of *DiffSeer*, and two open-ended questions (Q10, Q11) were used to gather suggestions from participants. Overall, the whole interview lasted about 60 minutes for each participant.

Results

Participants always got satisfactory answers for all tasks within 15 minutes, and we sum-

Table 1. The questionnaire for user interview. Q1-Q9 are closed-ended questions (Q1-Q4 for usability and Q5-Q9 for effectiveness). Q10-Q11 are open-ended questions.

ID	Questions
Q1	Is <i>DiffSeer</i> easy or hard to learn?
Q2	Is <i>DiffSeer</i> easy or hard to use?
Q3	Is the visual design easy or hard to understand?
Q4	Is the interaction helpful or not in the analysis process?
Q5	Is it easy or hard to understand differences across timeslices with the overview matrix?
Q6	Is it easy or hard to identify nodes with significant changes in the difference mask?
Q7	Is it easy or hard to check connection changes in each timeslice by unfolding the detail matrices?
Q8	Is the node reordering strategy helpful or not to enhance the visual pattern (like clustering) of the nested matrix?
Q9	Overall, is the proposed method helpful or not for identifying the temporal evolution of dynamic weighted graphs?
Q10	What are the advantages of the proposed method?
Q11	Which part of the method can be improved? How?

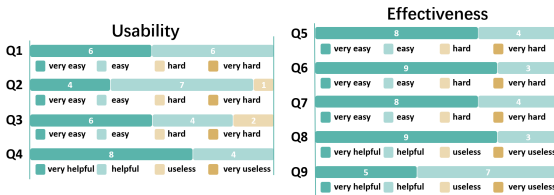


Figure 9. User interview results. Q1-Q4 is for usability, and Q5-Q9 is for effectiveness.

marized the participants’ standard of tasks as follows. Firstly, participants found the changes over time by observing the blue/red cells in the overview matrix (T1). They checked both overview and detail matrices and described the changes by the changed edge number and weight (T2). For T3, they found some repeated changes linked by difference mask. For T4 and T5, they reordered nodes and found nodes with similar changes gathering together.

Figure 9 summarizes the participants’ responses to our post-study questionnaire. Overall, both the usability and effectiveness of *DiffSeer* are highly rated by participants. They agree that the *DiffSeer* is convenient and efficient for exploring the temporal evolution of dynamic weighted graphs. However, two of them thought that the visual design might not be friendly for a novice to understand. The detailed comments from participants are summarized as follows:

Usability. All participants confirmed that they could easily learn and use *DiffSeer*, and the visual designs are intuitive. Some of them have used small-multiples or animation to explore dynamic weighted graphs before, and they confirmed that they can find changes of edge weight more quickly and accurately with *DiffSeer*. U1 mentioned that “*Since matrices are very common in graph analysis, I can easily understand the nested matrix design.*” U5 commented, “*It takes me some time to learn the meaning of Difference Mask, but it is quite useful after understanding it.*” In terms of the interaction, the participants appreciated the existing interactions of *DiffSeer*, which were confirmed to be beneficial and could satisfy the analysis requirements.

Effectiveness. Overall, all participants praised the effectiveness of *DiffSeer* in exploring the dynamic weighted graph evolution. They pointed out that the nested matrix design can help them quickly identify dynamic graph evolution characteristics, such as abrupt significant edge weight increases and recurrent edge weight changes. They said that it is more effective than the existing methods like small multiples. U3 pointed out that “*With the node reordering strategy, the detail matrix can be more helpful to confirm the graph details within one timeslice. But as for the weight of node reordering, I almost only chose 0 or 1 to focus on one matrix type since a median value is hard to interpret.*” However, U2 said that the adjustable weight feature helped him find a balance to analyze multiple matrices simultaneously.

Discussion

This section further discusses the generalizability, time range, and node number of *DiffSeer*.

Generalizability. We have shown the generalizability of *DiffSeer* by two case studies on real-world datasets with different characteristics, and the results can prove that *DiffSeer* is useful for different types of dynamic weighted graphs that need to be analyzed for changes. Our method can also support the dynamic unweighted graphs by considering the presence or absence of edges as an edge weight of 0 or 1. Although *DiffSeer* does not focus on the directed dynamic graphs, we can just encode one of the out-degree or in-degree in the overview matrix and reorder nodes

with inner-row similarity instead of the auto-correlation index, then *DiffSeer* can still work as expected.

Scalability. *DiffSeer* has good scalability in terms of the time range to be explored. The overview matrix can help explore more than one thousand timeslices simultaneously since the column width can be narrow enough as long as users can identify it. The time range can be further extended by brushing on the timeline view.

Conclusion and Future Work

We propose *DiffSeer*, a dynamic weighted graph visualization approach, by explicitly visualizing the differences on edge weight between adjacent timeslices, which incorporates a novel nested matrix design, a new node reordering strategy, and a rich set of interactions to help users fully understand the temporal evolution of dynamic graphs. For evaluation, we conduct two case studies on real-world datasets and in-depth interviews with 12 target users, and the results demonstrate that *DiffSeer* is useful and effective in visualizing dynamic graphs.

In future work, we plan to characterize differences from more perspectives, such as the removal or addition of nodes, to support broader analysis requirements. It is also interesting to further explore how the proposed difference-based visualization approach can be extended to the visualization of other datasets, like high dimensional time-series data.

REFERENCES

1. J.-w. Ahn, C. Plaisant, and B. Shneiderman. A task taxonomy for network evolution analysis. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):365–376, 2013.
2. B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J.-D. Fekete. Weighted graph comparison techniques for brain connectivity analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 483–492, 2013.
3. D. Archambault. Structural differences between two graphs through hierarchies. In *Proceedings of Graphics Interface*, pp. 87–94, 2009.
4. D. Archambault, H. Purchase, and B. Pinaud. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):539–552, 2010.
5. D. Archambault, H. C. Purchase, and B. Pinaud. Difference map readability for dynamic graphs. In *Proceedings of International Symposium on Graph Drawing*, pp. 50–61. Springer, 2010.
6. B. Bach, N. Henry-Riche, T. Dwyer, T. Madhyastha, J.-D. Fekete, and T. Grabowski. Small multiples: Piling time to explore temporal patterns in dynamic networks. *Computer Graphics Forum*, 34:31–40, 2015.
7. B. Bach, E. Pietriga, and J.-D. Fekete. Graphdiaries: Animated transitions and temporal navigation for dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):740–754, 2013.
8. B. Bach, E. Pietriga, and J.-D. Fekete. Visualizing dynamic networks with matrix cubes. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 877–886, 2014.
9. F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. *Computer graphics forum*, 36:133–159, 2017.
10. M. Behrisch, B. Bach, N. Henry Riche, T. Schreck, and J.-D. Fekete. Matrix reordering methods for table and network visualization. *Computer Graphics Forum*, 35:693–716, 2016.
11. T. Crnovrsanin, S. Chandrasegaran, K.-L. Ma, et al. Staged animation strategies for online dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):539–549, 2020.
12. J.-D. Fekete. Reorder.js: A javascript library to reorder tables and networks. In *Proceedings of IEEE VIS 2015*, 2015.
13. S. Rufiange and M. J. McGuffin. Diffani: Visualizing dynamic graphs with a hybrid of difference maps and animation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2556–2565, 2013.
14. S. Rufiange and G. Melançon. Animatrix: A matrix-based visualization of software evolution. In *Proceedings of 2014 IEEE Working Conference on Software Visualization*, pp. 137–146. IEEE, 2014.
15. P. M. Simon and C. Turkey. Hunting high and low: Visualising shifting correlations in financial markets. *Computer Graphics Forum*, 37:479–490, 2018.
16. P. Simonetto, D. Archambault, and S. Kobourov. Drawing dynamic graphs without timeslices. In *International Symposium on Graph Drawing and Network Visualization*, pp. 394–409. Springer, 2017.
17. P. Valdivia, P. Buono, C. Plaisant, N. Dufournaud, and J.-D. Fekete. Analyzing dynamic hypergraphs with parallel aggregated ordered hypergraph visualization. *IEEE*

Transactions on Visualization and Computer Graphics, 27(1):1–13, 2019.

18. N. van Beusekom, W. Meulemans, and B. Speckmann. Simultaneous matrix orderings for graph collections. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):1–10, 2021.
19. S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. Reducing snapshots to points: A visual analytics approach to dynamic network exploration. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):1–10, 2015.
20. J. S. Yi, N. Elmqvist, and S. Lee. Timematrix: Analyzing temporal social networks using interactive matrix-based visualizations. *Intl. Journal of Human–Computer Interaction*, 26(11-12):1031–1051, 2010.