Time-Aligned Edge Plots for Dynamic Graph Visualization

Moataz Abdelaal*, Antoine Lhuillier*, Marcel Hlawatsch*, and Daniel Weiskopf*
* VISUS, University of Stuttgart, Germany
Email: {moataz.abdelaal, marcel.hlawatsch, antoine.lhuillier, daniel.weiskopf}@visus.uni-stuttgart.de

***Abstract*—We present *time-aligned edge plots*: time- and edge-scalable representations of dynamic graphs. Vertices are mapped to two vertical parallel axes. The left axis depicts the source vertices, whereas the right one depicts the destination vertices. The time axis is horizontally embedded in-between the two axes, resulting in a two-dimensional graph layout. Edges are added by drawing straight lines connecting the corresponding source and destination vertices through time, while the pixels along the lines are used to encode the time-varying information. In this way, the depiction of edges at the individual timepoints is reduced to only a few pixels, resulting in a less cluttered representation of dynamic graphs, while the alignment of edges over time reveals the temporal patterns in the data and preserves the users' mental map. We evaluate our approach by comparing it theoretically and empirically against the state-of-the-art using dynamic graphs of varying complexities.**

***Keywords*-Visualization, dynamic graphs, time scalability, visual clutter**

I. Introduction

Visualizing large and dense static graphs is, by itself, a difficult task. Node-link diagrams or adjacency matrices are often used. Adding time as a third dimension increases the complexity of the data, and poses a visualization challenge, making even basic and essential tasks—such as getting an overview—challenging and non-trivial. Animation and small multiples are two popular approaches to map the time dimension. However, animation does not preserve the users' mental map, and small multiples are not scalable in time. In our work, we aim to provide a time-scalable overview of dynamic graphs that preserves the users' mental map and does not suffer from visual clutter.

While current visualization techniques [12], [37] are able to achieve time scalability, they suffer from overdrawing problems and, therefore, fail to attain edge scalability. This work contributes time-aligned edge plots (TEPs), a novel visualization approach that is scalable in the edge and time dimensions. Graph vertices are mapped to two parallel vertical axes representing the source and destination vertices. The time axis is horizontally embedded in-between the two axes, resulting in a two-dimensional layout. Edges are depicted by drawing straight lines connecting the source and destination vertices through time, while the strokes of the lines are drawn only at the respective timepoints where the edges occur. This allows us to depict the multiple occurrences of the same edge by only one line and, therefore, obtain a less cluttered visual representation of dynamic graphs.

II. Related Work

Static graphs build a complex data structure composed of vertices and edges. To visualize them, node-link diagrams or adjacency matrices are often used. Dynamic graphs add time as a third data dimension, posing a visualization challenge. In the recent decade, dynamic graph visualization became an active area of research. Beck et al. [9] discuss two major concepts to encode the time information into the graphs: animation (time-to-time mapping) [18], [19] and timeline (time-to-space mapping) [12], [37]. Animation bears a close resemblance to the real world. According to the Congruence Principle, it is an intuitive choice for conveying concepts of change [36]. However, when used for analysis tasks, animation may be too fast to be accurately apprehended and less effective than its static counterpart [34]. This is due to the limited capacity of humans to preserve information [3], [30] and detect changes (change blindness) [23] over time.

Alternatively, time-to-space mapping approaches use the screen space to encode the time dimension. A simple approach is to juxtapose the node-link diagrams in a small multiples fashion [11], [16]. To achieve the dynamic stability [18] between timepoints, graph layout algorithms can be employed [20], [26]. However, it is hard to trace the nodes across the timepoints, and the node-link graph layout is likely to be cluttered.

Other approaches attempt to achieve dynamic stability by transforming the node-link graph layout into a more structured layout where the positions of nodes are fixed over time, therefore, providing an effective way to compare the graphs at different points in time. The adjacency matrix layout [4], [33] is one example, the bipartite graph layout [13] is another. These approaches, however, are limited with respect to the number of individual graphs that can be displayed on the screen. Additionally, it requires the viewers to revisit the graphs back-and-forth to obtain an overview of the temporal changes, which is a challenging task given the limitations of humans' short-term memory [39] and perceptual issues like change blindness [23].

To address these problems, other approaches adopt a stacking representation to overlay small multiples on top of each other, resulting in a more condensed visualization where the changes between different timepoints can be instantly perceived, thus, reducing the users' cognitive load. Bach et al. stack the adjacency matrices on top of each other, resulting in a three-dimensional cube [5] or a pile [4] of adjacency matrices.

978-1-7281-9134-8/20/$31.00 ©2020 IEEE
DOI 10.1109/IV51561.2020.00048

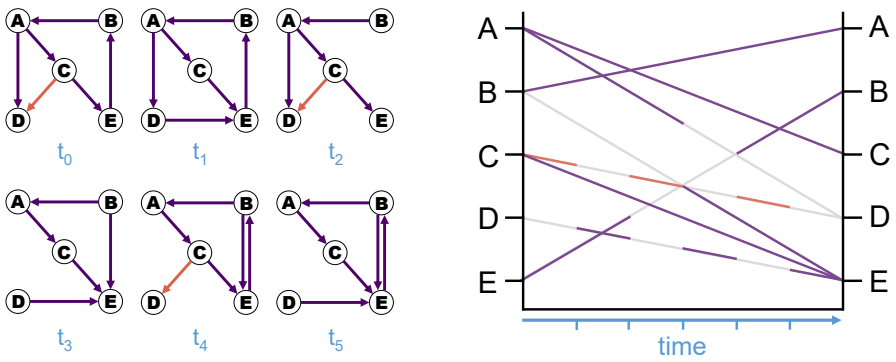


Fig. 1. Comparison between small multiples of node-link diagrams (left) and time-aligned edge plots (right). Edges over time are depicted by a single line in time-aligned edge plots.

Similarly, Abdelaal et al. [1] stack the individual bipartite graphs by drawing partial edges, resulting in a time-condensed visualization.

Although both approaches use the same stacking metaphor, the underlying graph layout affects the final result. The adjacency matrix layout is a two-dimensional representation. Adding the time dimension results in a three-dimensional representation, making it difficult to obtain an overview of the dynamic graph without applying interaction techniques. Moreover, the screen space consumed by the adjacency matrix grows quadratically with respect to the number of vertices, making these techniques only suitable for visualizing small graphs. The bipartite graph layout, in contrast, is based on a one-dimensional representation, making it more suitable for the stacking idea. However, since edges are drawn partially, it becomes more difficult to accurately determine the edge direction without the use of interaction.

Van den Elzen et al. [38] and Bach et al. [6] attempt to reduce the complexity of dynamic graphs by modeling the individual graphs as points in high-dimensional space and projecting them back to a two-dimensional node-link graph layout. However, in this case, the original graph's structure is not visible anymore, and the meaning of the resulting projection can be hard to grasp.

In our work, we aim to provide a static overview of dynamic graphs that is scalable in time and does not suffer from visual clutter. Instead of modeling a dynamic graph as a sequence of individual graphs at different points in time, we model the dynamic graph as a single super graph whose edges are changing over time. In this way, we avoid replicating the graph vertices at each timepoint, resulting in a time-scalable representation of dynamic graphs.

We situate our work close to the massive sequence views [37] and interleaved edge splatting [12]. Both approaches result in a two-dimensional visualization, where the x-axis encodes time, and the y-axis encodes the positions of vertices. Additionally, both employ vertex ordering to reduce the amount of visual clutter. Nevertheless, both approaches suffer from overdrawing problems caused by the overlapping edges that occur at the same timepoint in the massive sequence views, or by the interleaving of individual bipartite graphs in interleaved edge splatting. In our approach, regardless of how many timepoints the edge appears in, it will be depicted by a single line going from the source to the destination, while the pixels along the line are used to encode the time-varying information. In this way, the depiction of edges at the individual timepoints is reduced to just a few pixels, resulting in an edge-scalable visualization.

III. Method

We model a dynamic graph as a single super graph $G := (V, E)$ that consists of set vertices V and set of edges $E \subseteq V \times V$. This functional model allows us to depict each dynamic graph edge $e(v_i, v_j)$ by drawing a straight line going from the source v_i to the destination v_j through time T, while the varying width or color along the line encodes the edge weight $f_e(t) : \mathbb{R} \rightarrow \mathbb{R}$ at time $t \in [t_{\text{min}} \ldots t_{\text{max}}]$.

A. Layout

TEPs consist of one horizontal axis placed in-between two vertical parallel axes (see Figure 1 (right)). The horizontal axis depicts time, while the vertical axes depict graph vertices. The left vertical axis represents the source vertices, whereas the right one represents the destination vertices. To visualize dynamic graphs, vertices are placed on both vertical axes in the same order determined by our ordering algorithm (see Section III-D). Then, directed edges are drawn as straight lines from left to right to connect the corresponding source and destination vertices, through time. This results in a time- and edge-scalable representation of dynamic graphs.

Figure 1 shows a comparison between small multiples of a node-link graph layout (left) and TEPs (right). One can instantly perceive several temporal patterns in the data by looking at TEPs. For example, the periodic pattern of edges C→D and D→E, the stability of edges A→C and B→A, and the gaps in edges E→B and A→D. These patterns are difficult to obtain by looking at the small multiples, regardless of the underlying graph layout being employed (i.e., node-link graph layout, adjacency matrix, or bipartite graph layout). Typically,

it requires the viewers to revisit the individual graphs back-and-forth in order to see these patterns. Additionally, it is even more difficult to spot temporal synchronization such as the alternating behavior between the two edges C→D and D→E in the small multiples view. In contrast, in TEPs, the alignment of edges over time allows the temporal synchronization in the data to appear as spatial synchronization and, therefore easier to detect.

As shown in Figure 1, small multiples depict the graph structural information separately at each timepoint. In contrast, TEPs show the graph structure once for all timepoints. While increasing the number of timepoints results in new node-link diagrams in small multiples, it will only increase the width of a TEP by a few pixels. As shown in Figure 1, the edge C→D occurs at three timepoints (t_0, t_2, and t_4). In TEPs, these three occurrences are depicted by one line going from the C to D, such that the lines' stroke is drawn only at the respective timepoints when the edge occurs. This allows us to reduce the number of depicted graph lines by a significant amount, resulting in a less cluttered visual representation.

B. Edge Depiction

In TEPs, the depiction of edges depends on their occurrence frequency over time. While stable edges will be depicted as solid lines, periodic edges will appear as stippled or partially drawn lines (see Section III-E2). As a result, the visibility of an edge and, therefore, the ability to identify its source and target nodes correlates with the occurrence frequency of that edge. Burch et al. [14] found that drawing edges at 75% of their full length achieve the right balance between clutter reduction and perception of node connections. However, since edges can occur only for short periods (i.e., outliers), it becomes challenging to see them let alone trace the source and target nodes. To address this problem in TEPs, we draw reference lines for edges in the background. Instead of setting a fixed threshold for drawing the reference lines, we provide the users with adjustable settings. For example, the users can select to draw the reference lines for edges that have occurrence frequency between the minimum and the maximum of a defined interval. To distinguish between the reference lines in the background and the actual edges in the foreground, we increase the reference lines' transparency (see Figure 1 (right)).

To further amplify the recognition of edges in highly cluttered areas, we use color to encode the slopes of the lines. Additionally, we blend the lines by applying alpha compositing. As shown in Figure 2 (c), using both strategies increases the visibility of edges in crossing areas.

C. Time-Varying Attribute

Taking into account that the color is already used to encode the line slope, we experimented with three options to encode a time-varying attribute of the graph (i.e., edge weight): **(a)** Constructing a bi-variate color map from the uni-variate isoluminant color map [27]. This option allows us to use color to encode the edge weight besides the line slop (see Figure 3 (a)). **(b)** Using the line width as an additional variable

(a) (b) (c)

Fig. 2. By encoding the slope by color and applying alpha composition, we amplify the recognition of edges in highly cluttered areas. (a) Edges are drawn as opaque lines. (b) Edges are blended by applying alpha compositing. (c) Edges are blended, and slope is encoded by color.

to encode the edge weight. As illustrated in Figure 3 (b), a uni-variate color map is used to encode the slope, while the line width encodes the edge weight. **(c)** This option combines both options (a) and (b). In this way, the edge weight is encoded both by color and line width (see Figure 3 (c)).

Although option (a) works in principle, the use of color to encode quantitative data is not perceptually accurate [15], [28]—especially in cluttered areas of the graph, taking into account the blending of edges. Additionally, this option does not produce aesthetically pleasing results due to the low contrast of isoluminant colors. From our experiments, we found that options (b) and (c) provide a more accurate representation of edge weight with option (b) being more aesthetically pleasing due to the flexibility it offers when it comes to choosing the color map.

(a) (b) (c)

Fig. 3. Three options to encode a time-varying attribute while using the color to encode the line slope: (a) color encoding using bi-variate color map, (b) line width encoding, and (c) color encoding using bi-variate color map and line width.

D. Vertex Ordering

Since vertices are placed on the vertical axis, vertex-ordering algorithms play a critical role in obtaining a less cluttered visual representation. In graph theory, the problem of linearly ordering the vertices to optimize a cost function is an NP-hard problem, known as minimum linear arrangement [21], or MinLA problem. In our work, we combine hierarchical clustering with simulated annealing to obtain an efficient arrangement of vertices. Hierarchical agglomerative clustering is a deterministic algorithm except for the tied distances. The algorithm proceeds in a bottom-up approach by merging the two most similar clusters, until all clusters are hierarchically merged into one single cluster contains all graph vertices. The similarity between two graph vertices v_i and v_j is obtained by calculating the Jaccard coefficient:

$$J(\overline{V_i}, \overline{V_j}) = \frac{|\overline{V_i} \cap \overline{V_j}|}{|\overline{V_i} \cup \overline{V_j}|} \in [0, 1],$$

where $\overline{V_i}$ and $\overline{V_j}$ are the sets of direct neighbors for vertices v_i and v_j, respectively.

In other words, this similarity metric describes a measure for a common neighborhood of vertices. Vertices that have similar out- and in-connections will belong to the same cluster, and therefore, placed close to each other on the vertical axes. The resulting hierarchy is further reordered to minimize a cost function according to:

$$\text{minimize} : \bar{l}(\pi_V) = \frac{1}{|E|} \sum_{\forall e \in E} l(e), \tag{1}$$

where $\bar{l}(\pi_V)$ is the average edge length in the graph, π_V is a permutation on the set of vertices V, and $l(e)$ is the edge length defined as:

$$l(v_i, v_i) = \frac{|i - j|}{|V| - 1},$$

where i and j are the positions of v_i and v_j on the vertical axis in permutation π_V.

This allows us to further optimize the global connections between clusters without losing the hierarchical relationship between them. To do this, the hierarchy is traversed from top to bottom (breadth-first approach). For each cluster, the left and right children are swapped, and the cost function is computed before and after the swapping so that we keep the configuration that achieves minimum cost.

We further use the flexibility of simulated annealing to optimize the hierarchy resulting from the clustering algorithm, resulting in an even more optimized order of vertices. Similar to Van den Elzen et al. [37], we define a proportional cooling schedule, that is, $T_{i+1} = \alpha T_i$, where α is a constant close to but smaller than 1, known as the cooling factor. We chose a low value for the initial temperature ($T_0 = 100$) and cooling rate ($\alpha = 0.999$).

E. Visual Patterns

Applying a proper vertex-ordering technique and aligning the edge over time, allows TEPs to reveal some of the graph topological features and temporal patterns. In the following, we describe graph structural features and temporal patterns that can be recognized in TEPs.

1) Structural Patterns: In the following, we sketch some of the structural patterns that can be recognized in TEPs (see Figure 4).

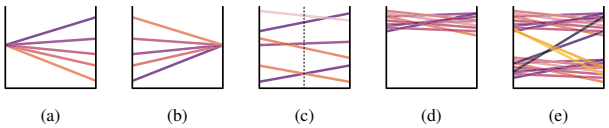


Fig. 4. Structural patterns in TEPs: (a) fan-out, (b) fan-in, (c) cross, (d) cluster, and (e) cross-cluster.

The **fan** pattern describes a single graph node that has a lot of incoming (fan-in) or outgoing (fan-out) connections. The connections are usually spreading in all directions, forming a fan-like shape (see Figure 4 (a) and (b)). If the same node has both fan-in and fan-out patterns, the overlapping between both patterns will form a diamond-like shape. The **cross** pattern describes a symmetrical relationship between two nodes, where the connections between them exist in both directions. This pattern forms a cross-like shape where the intersection point occurs exactly at the center of the graph (see Figure 4 (c)). This pattern does not apply if the intersection point occurs off-the-center of the graph. The **cluster** pattern describes a group of graph nodes that are strongly connected. In an ordered graph, the connections within a cluster are almost in a horizontal direction (see Figure 4 (d)). The **cross-cluster** pattern is a special case of the cross pattern, where the connections exist in both directions between two groups of nodes (two clusters), resulting in an hour-glass like shape (see Figure 4 (e)). This pattern does not require an equal number of connections between the two clusters.

2) Temporal Patterns: Similarly, we sketch some of the temporal patterns that could be recognized in TEPs (see Figure 5).

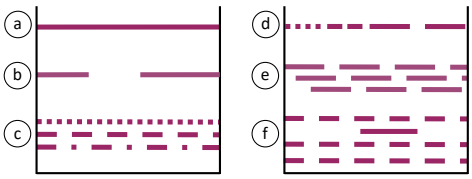
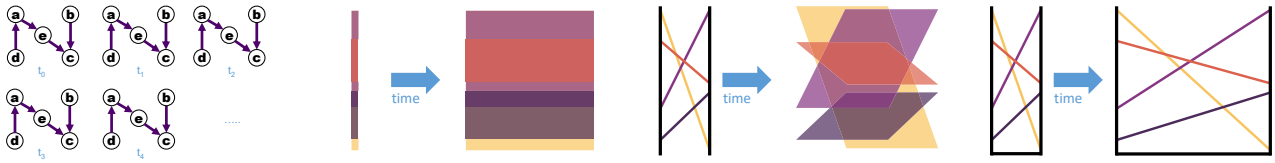


Fig. 5. Temporal patterns in TEPs: (a) stability, (b) gap, (c) periodicity, (d) trend, (e) phase shift, and (f) anomaly.

The **stability** pattern describes a constant presence of certain connection(s) over time. For a single connection, this pattern appears as a solid line going from the source to the destination (see Figure 5 (a)). The **gap** pattern describes a break in the continuity of a stable connection(s) for extended periods of time. For a single connection, this pattern appears as a solid line with a gap occurring at the beginning, the middle, or the end of it (see Figure 5 (b)). The **periodicity** pattern describes a regularly repeated presence and absence of certain connection(s) over time. For a single connection, this pattern appears as a dotted- or dashed-line. The lengths of the dashes or the gaps between the dots are not necessarily uniform over time (see Figure 5 (c)). A **trend** describes a gradual increase or decrease of certain connection(s) over time. For a single connection, this pattern appears as a dotted- or a dashed-line, where the lengths of the dashes or the gaps between the dots are gradually increasing or decreasing over time (see Figure 5 (d)). The **phase shift** pattern describes a series of two or more periodic connections that are regularly repeated in the same order over time. This pattern appears as two or more dotted- or dashed-lines synchronized together over time (see Figure 5 (e)). This pattern, by itself, is an analytical insight that could not be obtained easily using other visualization approaches. The **anomaly** pattern describes



(a) Small Multiples of NL (b) Massive Sequence Views (c) Interleaved Edge Splatting (d) Time-Aligned Edge Plots

Fig. 6. Two conceptual models for representing dynamic graphs. While small multiples approaches model the dynamic graph as a sequence on individual graphs, edge-based approaches model the dynamic graph as one super graph whose edges are changing over time.

an unusual presence of certain connections or a sudden change in their weights at specific points in time. This pattern appears as an unusual presence of a dot or a dash in a temporally coherent structure (see Figure 5 (f)). An anomaly timepoint is a point in time where the graph density remarkably deviates from the average density.

F. Zoom Lens

In TEPs, the information about edges direction and destination vertices are obtained by tracing the edges over the entire graph sequence. However, when the timepoints are not consistent with the surrounding context (i.e., anomaly timepoints), it is hard to obtain such information. To address this problem, we implemented an interactive nonlinear time-zooming lens that allows the users to magnify the timepoints by moving the mouse cursor over them. To maintain the slope information unchanged, we perform nonlinear scaling of time: timepoints near the center of the lens are enlarged, whereas the ones near the borders of the lens are shrunk in size. As shown in Figure 12, zooming at the anomaly timepoint allows us to obtain more information about the edges that occur at it, while at the same time preserving the slope information inside and outside the zooming lens.

IV. Evaluation

To evaluate our approach, we compared it against two state-of-the-art techniques that share several aspects with TEPs: massive sequence views (MSVs) [37] and interleaved edge splatting (IES) [12]. The evaluation is composed of a theoretical comparison (presented in Section IV-A) complemented by an empirical one (presented in Section IV-D). The empirical assessment is done by qualitative results inspection (QRI) of visual results, a frequently used evaluation method in visualization research [24]. To assure the rigorousness of the QRI and avoid cherry-picking the datasets [35], we synthetically generated dynamic graph datasets with varying complexities based on a scale-free network generation model (see Section IV-B). Furthermore, we selected three analysis tasks to serve as a base for our inspection (see Section IV-C). Finally, in Section V, we further asses the capability of TEPs by visualizing a real-world network that does not share the scale-free property.

A. Theoretical Comparison

In the following, we distinguish between the conceptual models behind the three visualization techniques in contrast to the conceptual model behind the traditional approaches of small multiples. Then, we comment on three technical aspects; the depiction of edges, encoding of color and slope, and the graphs' structural patterns. Due to the space limitation, we refrain ourselves from describing the techniques in detail. The interested reader, however, is encouraged to read the respective publications.

1) Conceptual Model: TEPs, MSVs, and IES are edge-based visualization techniques. Edge-based techniques model the dynamic graph as one super graph whose edges are changing over time. Therefore, graph vertices are drawn only once for the entire sequence, while edges are repeated at each timepoint. In contrast, small multiples approaches model the dynamic graph as a sequence of individual graphs. Therefore, graph vertices and edges are drawn repeatedly at each timepoint (see Figure 6).

This distinction in modeling allows edge-based techniques to be scalable with respect to the time dimension more than small multiples approaches, and, as a result, providing a better overview of the temporal events. In contrast, in edge-based techniques, changes in graph vertices are not expressed by the vertices themselves, but rather by the edges attached to them. Therefore, temporal changes related to the graph vertices are not easily recognized in these techniques (i.e., node addition/removal).

2) Edge Depiction: The difference between TEPs and the other edge-based approaches lies in the way the edges are depicted over time. As seen in Figure 6, in both IES and MSVs, edges are depicted by drawing straight lines going from the source to the destination, resulting in a sequence of lines over time. These sequences of lines form block structures that cover certain areas of the graph, depending on the edge length and time span. In large dynamic graphs, these blocks are likely to be drawn on top of each other, resulting in a cluttered visual representation, where the temporal patterns are hardly seen. In contrast, in TEPs, each edge is depicted by a single line throughout the entire graph sequence. This reduces the number of depicted lines by a significant amount, resulting in an edge-scalable visualization and revealing temporal patterns in the graph.

3) Color and Slope Encoding: In TEPs and IES, the depiction of edges essentially relies on the edge slope or inclination. To amplify the recognition of slope in highly cluttered areas, color is used to encode this information. In contrast, in MSVs, since all edges are drawn as vertical lines,

the slope encoding is irrelevant. Hence, the technique makes use of color to encode the length and direction (top or down) of edges. However, relying on color alone to distinguish edges might be problematic because the color of an edge is, actually, the alpha composition of all edges underneath. Therefore, it is highly likely that the edge will have different colors as the density of the graph changes over time. While this problem is present in the three visualization techniques, it is less critical in TEPs, due to the small amount of overdrawing and the slope encoding, and severe in MSVs, due to the large amount of overdrawing and the absence of slope encoding.

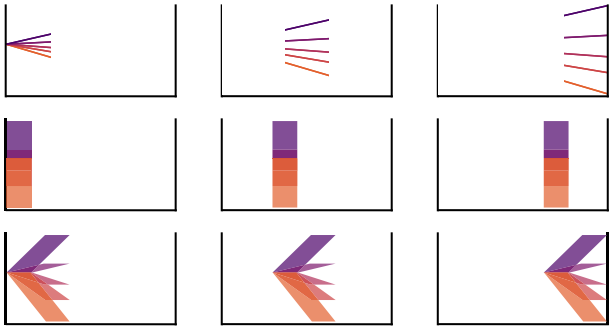


Fig. 7. Fan-out pattern depicted over time in TEPs (top), MSVs (middle), and IES (bottom). While the shape of the pattern remains unchanged in MSVs and IES, it is gradually changing in TEPs.

4) Structural Patterns: Compared to IES and MSVs, the graphs' structural information in TEPs is depicted over the entire graph sequence. Therefore, the same graph structure might appear differently, depending on the point in time it occurs. Figure 7 shows how the fan-out pattern is depicted over time in each visualization approach. While the shape of the pattern remains unchanged in MSVs and IES, it is gradually changing in TEPs, affecting the recognition of the same pattern over time.

B. Generative Data Model

To generate synthetic dynamic graphs, we adopted the model proposed by Cooper et al. [17]. This model extends the preferential attachment model introduced by Barabási and Albert [8] by considering four elementary processes that are present in many real networks: the addition and removal of edges and vertices, over time. While the addition processes follow the preferential attachment model, the removal is done rather randomly. By suitable choice of the input parameters, the model produces scale-free graphs, where the degree distribution of vertices follows the power-law $p_k \sim k^{-\gamma}$, a property that has been observed in many real-world networks [7], [31].

The generative model has the following input parameters:

1) $\alpha, \alpha_0, \alpha_1$: the input parameters in the original model. The choice of these parameters has to satisfy the constraints mentioned in the proposed model [17]. Generally, α has to satisfy the condition $\alpha > 0.5$. Choosing $\alpha_1 > \alpha_0$, assigns more weight to the probabilities of node events. In contrast, $\alpha_0 > \alpha_1$, assigns more weight to the probabilities of edge events.
2) m: defines how many edges to be added/removed randomly at each timepoint. It also defines the number of neighbors for the new nodes
3) t: defines the number of timepoints (iterations).

Similar to Okoe et al. [32], we adopted a linear density definition as $d_\ell = |E|/|V|$, which considered a better descriptor of the complexity of the real-world networks [29]. Since the parameter m defines how many edges to be added or deleted at each iteration, we can maintain a relatively stable edge density over time by setting the m parameter to the desired density accompanied by suitable α's probabilities. The graph size n, in contrast, is controlled indirectly by the number of timepoints t and the value of the α_1 parameter. Setting α_1 close or equal to 1.0 increases the probability of adding a new node at each iteration, implying that the total size of the graph will be close to the number of timepoints.

By manipulating the m, t, and α parameters, we can generate dynamic graphs of varying complexities. We experimented with three density values $m = \{1, 2, 4\}$, to generate dynamic graphs where the number edges is {equal, double, or four times} the number of nodes. By choosing these values, we aim to capture the average density found in many real-world networks [29], [32]. Similarly, we experimented with two sets of α parameters $\{\alpha = 0.85, \alpha_0 = 0.01, \alpha_1 = 0.75\}$ and $\{\alpha = 0.6, \alpha_0 = 0.35, \alpha_1 = 0.2\}$. By the first and second sets, we aim to evaluate the techniques against node-related and edge-related events, respectively. Both sets result in scale-free graphs with degree exponent $\gamma \approx 3$, which lies in the range $2 \leq \gamma \leq 4$. This range has been identified as the degree exponent range for most real-world networks [7], [22].

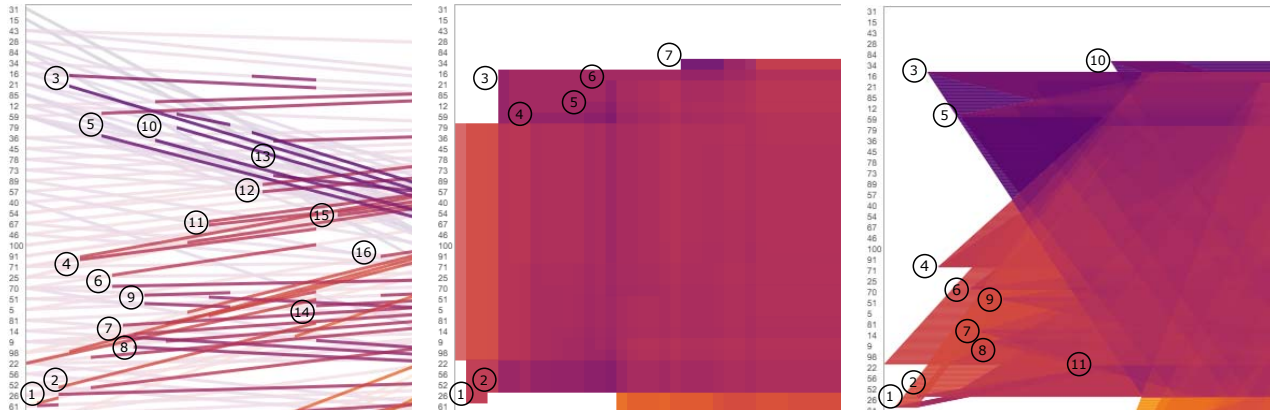
We decided to use fixed number of timepoints $t = 100$. On the one hand, we wanted to narrow the scope of our evaluation to be only against the edge density, which is the critical factor, considering the techniques under evaluation. On the other hand, we wanted to ensure that the resulting visualizations could fit nicely in a browser. This results in a total of six dynamic graphs listed in Table I. For simplicity, we assume that all edges have the same weight (i.e., unweighted graphs).

TABLE I
BY EXPERIMENTING WITH THREE DENSITY VALUES AND TWO DIFFERENT SETS OF α PARAMETERS, WE PRODUCE SIX DIFFERENT GRAPHS OF VARYING COMPLEXITIES (G1 – G6).

	Edge density		
	1	2	4
Node-related events	G1	G2	G3
Edge-related events	G4	G5	G6

C. Analysis Tasks

We assume an explorative analysis scenario where the analyst has no prior knowledge of the data and wants to see *what* is happening in the network? *When* does it happen? And *how* does it take place? To decide upon the analysis tasks that are



(a) TEPs (b) MSVs (c) IES

Fig. 8. A cut-off of graph G2 visualized using TEPs, MSVs, and IES. More node addition events are identified in TEPs.

suitable for answering these high level questions, we considered the task taxonomy for network evolution analysis [2]. Based on that, we chose the following analysis tasks as a base for the evaluation process:

T1: Identifying node addition/removal events
T2: Identifying link addition/removal events
T3: Identifying temporal patterns (see Section III-E2)

Since the node-related events are always expressed by links, edge-based visualization approaches suffer from an *ambiguity problem* when it comes to identifying the moments of node birth/death (Task T1). One way to tackle such a problem is by determining the first/last occurrence of a link event(s) where the node under question was involved. Due to the generative model, each new node will be connected to m existing nodes at the moment of birth. Therefore, the task of identifying newly added nodes, is, in fact, a task of identifying m links coming out from the same node at the same point in time (i.e., fan-out pattern). Similarly, to identify removed nodes we were looking for multiple links removed at the same point in time. If the target node is removed, the event appears as an incomplete fan-in pattern. In contrast, if the source node is removed, the event appears as an incomplete fan-out pattern. The more connections the node has, the more prominent will be the event.

Nevertheless, finding these first/last-occurred fan patterns does not necessarily mean that the node was added/removed at this timepoint. It is only a likelihood that correlates with the number of links. For example, if a node was inactive from the beginning of the graph sequence, and suddenly many links, involving this node, start to occur precisely at the same timepoint. Then, it is highly likely that the node was created/added at this timepoint. This is a special case due to the way edge-based techniques model and represent dynamic graphs and the generative data model being used.

Depending on the technique being evaluated, some of these tasks might require an additional step(s) from the analyst. For example, identifying the source and target nodes of a link requires mapping links end-points to the vertical axes. In MSVs and IES, links are drawn in a full length at each timepoint. Hence, identifying the source and target nodes is done by merely projecting the two end-points to the vertex axes. In contrast, in TEPs, the identification is done by extending the two end-points along the line slope, until they meet the two vertex axes. However, it might be hard to determine the slope of the links that occur for a short period, without drawing reference lines in the background.

D. Discussion of the Results

We generated six random graphs (see Table I) using the generative model described in Section IV-B. We then visualized those graphs by the three visualization techniques. In this section, we summarize the results of our qualitative comparison based on the fore mentioned analysis tasks. For the node-related tasks, we considered graphs G1, G2, and G3. For the link and temporal patterns-related tasks, we considered graphs G4, G5, and G6. For illustration purposes and due to the space limitation, we only show selected examples by cutting off the resulting images. To view the complete set of images in the original size, please refer to the supplementary materials[1].

1) T1: Identifying Node Addition/Removal Events: We were able to identify more node events in TEPs, in comparison to MSVs and IES. Figure 8 shows a cut-off graph G2 visualized using TEPs (a), MSVs (b), and IES (c). We identified 16, 11, and 7 node addition events in TEPs, IES, and MSVs, respectively. TEPs and IES both utilize the line slope as an additional variable for visualizing links. However, TEPs shows better results when the out-going links have similar slopes, for example, events 11, 12, and 15 in Figure 8 (a). In contrast, links with similar slopes are drawn over each other in IES, making it difficult to see the fan-out pattern, and therefore, hiding the node addition event. In MSVs, to identify new nodes, we look for the rectangular corners (see Figure 8 (b)). However, since links do not have a slope, we cannot distinguish between a node addition event, link addition event, or random intersections between rectangles. Although TEPs provide a less cluttered visualization compared to MSVs and IES, the visibility of the

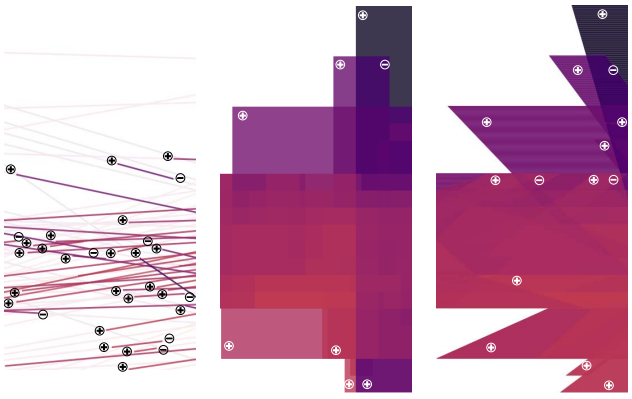
[1]https://doi.org/10.5281/zenodo.3975572

node addition event depends on the density of the graph. The denser the graph, the less visible are the events.

Similar to node addition events, node removal events are more visible in TEPs. For example, in graph G2, we detected six node removal events in TEPs. In MSVs and IES, we were able to spot only two of them. The remaining events appear as a slight change in the color opacity, which could be easily overlooked or miss-interpreted without prior knowledge.

2) T2: Identifying Link Addition/Removal events: Similar to node events, we were able to identify more link events in the resulting images from TEPs, in comparison to MSVs and IES. To have a concise comparison, we inspected a sequence of 34 timepoints in the middle of the graph sequence G4, from timepoint `time_36` to timepoint `time_70` (see Figure 9). In TEPs, we identified more than 20 added links (annotated by ⊕) and eight removed links (annotated by ⊖). In MSVs and IES, while we recognized some of the added links (seven in MSVs, and eleven in IES), it was particularly hard to spot the deleted ones. We only saw three deleted links in IES and one in MSVs.

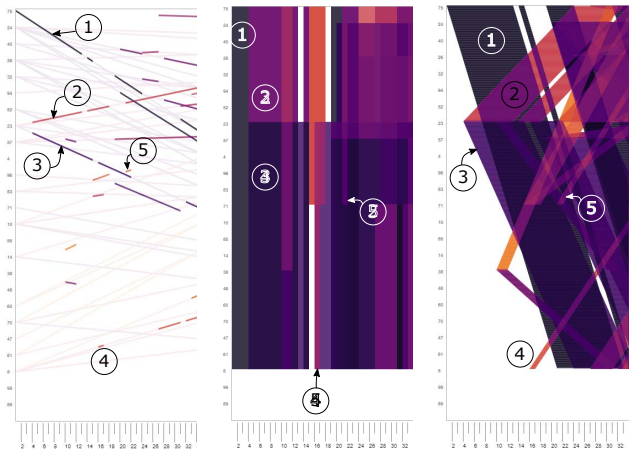
As the density increases in graph G6, it becomes more challenging to detect link events. While this holds for all techniques, TEPs shows better results compared to the others. In TEPs, link events appear as a presence or absence of lines. In contrast, in MSVs and IES, link events appear as a change in color opacity, making them easy to miss or overlook without prior knowledge.



(a) TEPs (b) MSVs (c) IES

Fig. 9. A cut-off of graph G4 between timepoint `time_36` and timepoint `time_70` visualized using TEPs, MSVs, and IES. More link events are identified in TEPs.

3) T3: Identifying Events Temporal Patterns: When it comes to identifying the temporal patterns of events, TEPs provide a better overview of such patterns. Figure 10 shows the first 33 timepoints of graph G5 visualized using TEPs, MSVs, and IES. We annotated some of the temporal patterns that we could recognize in each technique. In TEPs, we identified a periodic behavior for four links: `75->8`①, `23->75`②, `23->8`③, and `8->71`④. Additionally, we spotted one link that lasts for exactly one timepoint: link ⑤ at timepoint 21. However, it is hard to determine the source and target nodes for this link without adjusting drawing settings for the reference lines.



(a) TEPs (b) MSVs (c) IES

Fig. 10. A cut-off of the first 33 timepoints from graph G5 visualized using TEPs, MSVs, and IES. Links ①, ②, ③, and④ exhibit a periodic behavior.

In MSVs, it is challenging to spot links with periodic behavior. This is because links occurring at the same timepoint will be drawn over each other, and the final color of the top link will be the accumulated color of all links underneath. As a result, it is hard to recognize the same link over time, due to the color change. As seen in Figure 10 (a), the first occurrence of link `75->8`① took place between timepoints 1 and 13. However, looking at the resulted image from MSVs in Figure 10 (b), we could recognize link ① up until timepoint 4. After that, the color of the link changes due to the appearance of other links. The same thing can be noticed in links `23->75`② and `23->8`③. This change in color misleads our perception of identifying periods of time where an edge persisted, making the detection of periodicity patterns very challenging.

Although IES suffers from the same overdrawing problem as MSVs, the usage of the slope encoding allows us to still recognize the links despite the color change. For example, we recognized the periodicity of links `75->8`①, `23->75`②, and `8->71`④. However, we cannot see the patterns easily if the links have similar slopes. For example, `23->8`③ is overlapping with `75->8`①. Since both links have similar slopes, they have similar colors, making it hard to distinguish between them in dense areas. As opposite to TEPs, we identified the source and target nodes of link ⑤ that occur at timepoint 21, link `71->24`. This is an advantage of IES and MSVs since links are drawn in full length, irrespective of how long they persisted. In IES, due to the interleaving, links end-points are not perfectly aligned in time. While the start point corresponds to the timepoint where the link occurs, the end point is shifted in time, depending on the stripe width. This could be misleading sometimes.

V. Real-world Example: Software Call Graph Dataset

Several real-world networks, such as neural networks or power grids, are not scale-free networks [7]. In this section, we further assess the capabilities of TEPs by visualizing a

real-world network that does not share the scale-free property. The software call graph dataset is the dynamic method calls of an open-source Java system called JHotDraw [25]. We chose to visualize the dynamic method calls resulting from a drawing scenario done by Beck at al. [10] using the JHotDraw graphical editor (JavaDrawApp). The test scenario was to start the program, create a new file, draw a rectangle, draw a circle, and write a text into the circle. The resulted dynamic method calls from this scenario were used to evaluate different dynamic graph visualization techniques [1], [10], [12]. The dataset contains 787 vertices and 1077 timepoints, and has an average edge density ≈ 0.91.

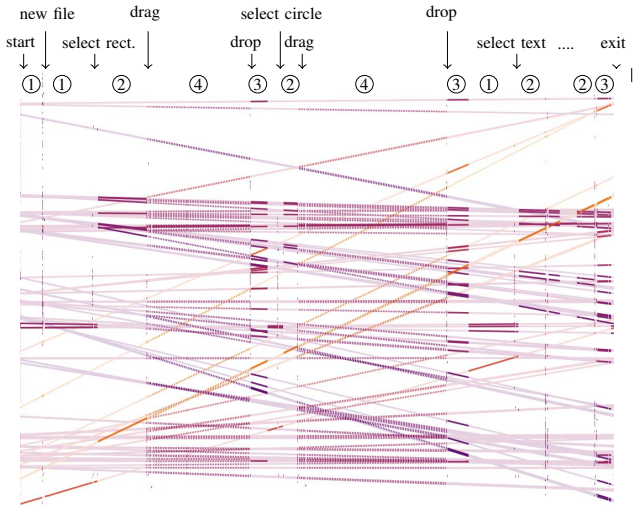


Fig. 11. Software call graph dataset visualized using TEPs. Four different temporal patterns can be identified.

Figure 11 shows the dynamic method calls of the drawing scenario visualized using TEPs. For the resulting images of MSVs and IES, please refer to the supplementary materials. At the top of the figure, we annotated the timeline with the corresponding user actions. By inspecting the resulted graph, we can clearly distinguish between four different temporal patterns. Each of the identified patterns depicts functions calls that are triggered as a reaction to the user command. Beck at al. [10] provided some interpretations for these patterns. For example, Pattern 1 reflects the function calls that are triggered when the user moves the mouse over the toolbar. Pattern 2 and Pattern 3 reflect the mouse movement over the drawing canvas. In Pattern 2, the mouse moves over an empty canvas, while in Pattern 3, it moves over the drawn objects. Pattern 4 reflects the function calls when the user draws (drags) the object. In this way, one can tell the drawing scenario by visually following the user actions and function calls patterns, as depicted in Figure 11.

Each of the user actions annotated in Figure 11 at the top results in a dense timepoint, where many function calls occur. A typical example is the first timepoint when the program starts. These are probably functions calls related to the program's initialization process. Another example is when the user creates a new file. From a graph point of view, these timepoints are

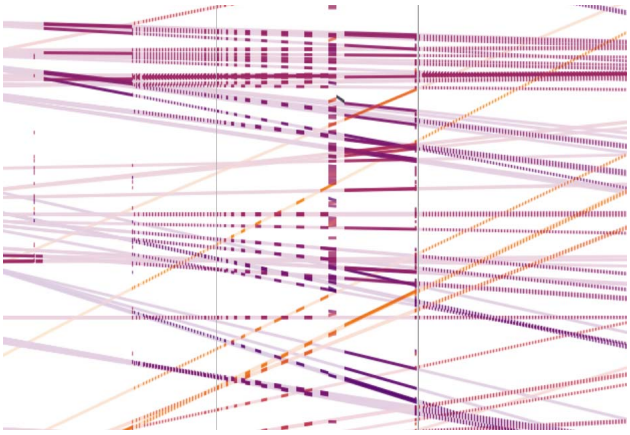


Fig. 12. Zooming in the timepoints allows us to obtain more information without losing the context information such as edge direction or slope.

considered outliers compared to the whole graph sequence. We use the zoom-in lens to magnify these timepoints without losing the context information such as edge direction or slope. In Figure 12, we show a zoomed-in version of the outlier timepoints correspond to the mouse-drop action performed by the user after drawing the rectangle and the circle. By inspecting both timepoints, we can see that they are sharing the same set of function calls. However, we notice some of the edges appear in different spatial locations in both timepoints, since TEPs does not preserve the shape of the graphs' structural pattern over time. This requires an additional effort to trace the edges by following their slopes.

VI. Conclusion

In this paper, we introduced time-aligned edge plots (TEPs), a representation of dynamic graphs that is scalable in the time and edge dimensions. The graph vertices are mapped to two vertical parallel axes representing the source and destination vertices. The time dimension is horizontally embedded between the two axes, resulting in a two-dimensional layout. Edges are depicted by lines connecting the source and destination vertices, through time, while the line-based graphical primitives are used to encode the time-varying attribute(s). This allows us to significantly reduce the number of pixels used to depict the edges at the individual timepoints, and, therefore, obtaining a less cluttered representation.

To evaluate our approach, we compared it against massive sequence views (MSVs), and interleaved edge splatting (IES) using dynamic graphs of varying complexities. We showed that TEPs obtain good results in identifying edge-related events and graphs' temporal patterns. However, due to the partial drawing of edges, it might be hard to determine the source and target nodes of certain events. Nevertheless, the technique provides a less-cluttered overview of the temporal changes, which is lacking in the other approaches. This suggests that TEPs can serve as an entry point for analyzing networks' temporal features, while other traditional visualization approaches could be integrated to obtain detailed insights.

ACKNOWLEDGMENT

We thank Wladimir Ponomarenko for implementing and testing earlier proof-of-concept versions of the visualization technique. We also thank the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) for supporting this work under Germany's Excellence Strategy – EXC 2120/1 – 390831618.

REFERENCES

[1] M. Abdelaal, M. Hlawatsch, M. Burch, and D. Weiskopf, "Clustering for stacked edge splatting," in *Proceedings of the Conference on Vision, Modeling, and Visualization (VMV)*, 2018, pp. 127–134.

[2] J. Ahn, C. Plaisant, and B. Shneiderman, "A task taxonomy for network evolution analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 3, pp. 365–376, 2014.

[3] D. Archambault, H. C. Purchase, and B. Pinaud, "Animation, small multiples, and the effect of mental map preservation in dynamic graphs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 4, pp. 539–552, 2011.

[4] B. Bach, N. Henry-Riche, T. Dwyer, T. Madhyastha, J.-D. Fekete, and T. Grabowski, "Small MultiPiles: piling time to explore temporal patterns in dynamic networks," *Computer Graphics Forum*, vol. 34, no. 3, pp. 31–40, 2015.

[5] B. Bach, E. Pietriga, and J. Fekete, "Visualizing dynamic networks with matrix cubes," in *Proceedings of Conference on Human Factors in Computing Systems (CHI)*, 2014, pp. 877–886.

[6] B. Bach, C. Shi, N. Heulot, T. Madhyastha, T. Grabowski, and P. Dragicevic, "Time curves: Folding time to visualize patterns of temporal evolution in data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 559–568, 2016.

[7] A.-L. Barabási, *Network Science*. Cambridge: Cambridge University Press, 2016.

[8] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[9] F. Beck, M. Burch, S. Diehl, and D. Weiskopf, "A taxonomy and survey of dynamic graph visualization," *Computer Graphics Forum*, vol. 36, no. 1, pp. 133–159, 2017.

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

[11] I. Boyandin, E. Bertini, and D. Lalanne, "A qualitative study on the exploration of temporal changes in flow maps with animation and small-multiples," in *Computer Graphics Forum*, vol. 31, no. 3, 2012, pp. 1005–1014.

[12] M. Burch, M. Hlawatsch, and D. Weiskopf, "Visualizing a sequence of a thousand graphs (or even more)," *Computer Graphics Forum*, vol. 36, no. 3, pp. 261–271, 2017.

[13] M. Burch, C. Vehlow, F. Beck, S. Diehl, and D. Weiskopf, "Parallel edge splatting for scalable dynamic graph visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2344–2353, 2011.

[14] M. Burch, C. Vehlow, N. Konevtsova, and D. Weiskopf, "Evaluating partially drawn links for directed graph edges," in *Graph Drawing*, M. van Kreveld and B. Speckmann, Eds. Berlin, Heidelberg: Springer, 2012, pp. 226–237.

[15] W. S. Cleveland and R. McGill, "Graphical perception and graphical methods for analyzing scientific data," *Science*, vol. 229, no. 4716, pp. 828–833, 1985.

[16] C. S. Collberg, S. G. Kobourov, J. Nagra, J. Pitts, and K. Wampler, "A system for graph-based visualization of the evolution of software," in *Proceedings ACM 2003 Symposium on Software Visualization*, 2003, pp. 77–86, 212–213.

[17] C. Cooper, A. Frieze, and J. Vera, "Random deletion in a scale-free random graph process," *Internet Mathematics*, vol. 1, no. 4, pp. 463–483, 2004.

[18] S. Diehl and C. Görg, "Graphs, they are changing," in *Proceedings of the Symposium on Graph Drawing*, 2002, pp. 23–30.

[19] Y. Frishman and A. Tal, "Dynamic drawing of clustered graphs," in *Proceedings of 10th IEEE Symposium on Information Visualization*, 2004, pp. 191–198.

[20] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software: Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

[21] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman, 1979.

[22] G. Ghoshal, L. Chi, and A.-L. Barabási, "Uncovering the role of elementary processes in network evolution," *Scientific Reports*, vol. 3, no. 2920, 2013.

[23] C. G. Healey and J. T. Enns, "Attention and visual memory in visualization and computer graphics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 7, pp. 1170–1188, 2012.

[24] T. Isenberg, P. Isenberg, J. Chen, M. Sedlmair, and T. Mller, "A systematic review on the practice of evaluating visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2818–2827, 2013.

[25] "JHotDraw Start Page," http://www.jhotdraw.org/, 2018, accessed on 03/22/2018.

[26] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Information Processing Letters*, vol. 31, no. 1, pp. 7–15, 1989.

[27] P. Kovesi, "Good colour maps: How to design them," *arXiv preprint arXiv:1509.03700*, 2015.

[28] J. Mackinlay, "Automating the design of graphical presentations of relational information," *ACM Transactions on Graphics*, vol. 5, no. 2, pp. 110–141, 1986.

[29] G. Melancon, "Just how dense are dense graphs in the real world?: A methodological note," in *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization (BELIV)*, 2006, pp. 1–7.

[30] K. Misue, P. Eades, W. Lai, and K. Sugiyama, "Layout adjustment and the mental map," *Journal of Visual Languages and Computing*, vol. 6, no. 2, pp. 183–210, 1995.

[31] M. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.

[32] M. Okoe, R. Jianu, and S. Kobourov, "Revisited experimental comparison of node-link and matrix representations," in *Graph Drawing and Network Visualization*, F. Frati and K.-L. Ma, Eds. Cham: Springer International Publishing, 2018, pp. 287–302.

[33] A. Perer and J. Sun, "MatrixFlow: temporal network visual analytics to track symptom evolution during disease progression," in *AMIA Annual Symposium Proceedings*, vol. 2012. American Medical Informatics Association, 2012, pp. 716–725.

[34] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko, "Effectiveness of animation in trend visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1325–1332, 2008.

[35] C. Schulz, A. Nocaj, M. El-Assady, S. Frey, M. Hlawatsch, M. Hund, G. K. Karch, R. Netzel, C. Schätzle, M. Butt, D. A. Keim, T. Ertl, U. Brandes, and D. Weiskopf, "Generative data models for validation and evaluation of visualization techniques," in *Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization, BELIV 2016*. ACM, 2016, pp. 112–124.

[36] B. Tversky, J. B. Morrison, and M. Betrancourt, "Animation: can it facilitate?" *International Journal of Human-Computer Studies*, vol. 57, no. 4, pp. 247–262, 2002.

[37] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk, "Reordering massive sequence views: Enabling temporal and structural analysis of dynamic networks," in *Proceedings of IEEE Pacific Visualization Symposium*, 2013, pp. 33–40.

[38] ——, "Reducing snapshots to points: A visual analytics approach to dynamic network exploration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 1–10, 2016.

[39] C. Ware, *Information Visualization: Perception for Design*. San Francisco: Morgan Kaufmann, 2004.