GraphDiaries: Animated Transitions and Temporal Navigation for Dynamic Networks

Benjamin Bach, Emmanuel Pietriga, Jean-Daniel Fekete

GraphDiaries: Animated Transitions and Temporal Navigation for Dynamic Networks.

Benjamin Bach, Emmanuel Pietriga, and Jean-Daniel Fekete, *Senior Member, IEEE*

Abstract—Identifying, tracking and understanding changes in dynamic networks are complex and cognitively demanding tasks. We present *GraphDiaries*, a visual interface designed to improve support for these tasks in any node-link based graph visualization system. GraphDiaries relies on animated transitions that highlight changes in the network between time steps, thus helping users identify and understand those changes. To better understand the tasks related to the exploration of dynamic networks, we first introduce a task taxonomy, that informs the design of GraphDiaries, presented afterwards. We then report on a user study, based on representative tasks identified through the taxonomy, and that compares GraphDiaries to existing techniques for temporal navigation in dynamic networks, showing that it outperforms them in terms of both task time and errors for several of these tasks.

Index Terms—Dynamic Networks, Graph Visualization, Temporal Navigation, User experiment.

1 Introduction

In recent years, there have been many advances in the domain of network visualization, ranging from novel methods to improve their visual representation, to elaborate interaction techniques that ease navigation and exploration. However, these advances have mostly targeted static networks, even though most networks are dynamic in nature: social networks, business networks, communication and computer networks.

The processes underlying network evolution are hard to understand, and add an additional level of complexity to network analysis. In dynamic networks, nodes and links appear, possibly disappear, and sometimes re-appear. These low-level changes are responsible for higher-level changes, such as the emergence of central actors, or the merging of two clusters. In contrast to general graph metrics such as density or diameter that can easily be plotted over time in a simple chart, those low-level changes cannot be visualized so easily.

To explore dynamic networks, current visualization systems either a) aggregate information about time and changes in one single image, b) employ a three-dimensional visualization based on the space-time cube metaphor, c) represent graph time steps as series of juxtaposed images (space-multiplex), or d) display the network one step at a time, sometimes providing animations in-between (time-multiplex). Aggregated, 3D and juxtaposed images are useful but limited, in the sense that they do not scale well with the number of time steps, network size and number of changes. Despite their relative simplicity, they can prove difficult to integrate in existing network visualization systems, as they possibly require major modifications to the underlying visual interface.

- *B. Bach and J.-D. Fekete are with INRIA, Saclay, France. E-mail: benjamin.bach@inria.fr, jean-daniel.fekete@inria.fr*
- *E. Pietriga is with INRIA, Saclay, France and INRIA Chile – CIRIC, Santiago, Chile. E-mail: emmanuel.pietriga@inria.fr*

Time-multiplex offers several advantages over the other, more static representations, and turns the problem of visualizing temporal information (*when, how long, how often*, etc.) into actively navigating the network for understanding changes. Showing each time step in a separate image reduces visual complexity, as only the nodes and edges actually present at a given time step have to be shown in the corresponding image. Each stage in the graph's evolution can be observed independently, displayed using any static network visualization method, thus enabling the representation of domain-specific information about nodes and edges or network analysis metrics, such as node centrality or group membership. Time multiplex being relatively independent from the number and granularity of time steps, users navigate between time steps and observe changes across single images, possibly supported by animated transitions. Animations can provide some cognitive support to users trying to relate different steps, for instance to indicate state changes [1]. However, their value tends to decrease as the number of elements that change between two steps increases. Another limitation comes from their inability, in their basic form, to transition between non-contiguous time steps. All animated transitions between intermediate steps have to be played, which makes the comparison of distant time steps difficult, actually increasing users' cognitive load.

This article introduces techniques to improve temporal navigation in dynamic networks, focused on providing a higher level of flexibility and better support for exploring changes between steps. We investigate how staged animations which change highlighting and complementary small multiples help to understand changes between individual time steps while users freely navigate the dynamic network. Our main contributions are:

- A simple yet expressive taxonomy to describe low-level and higher-level tasks associated with the exploration of dynamic networks, along three dimensions *time* (when), *graph elements* (where), and *type of change* (what).

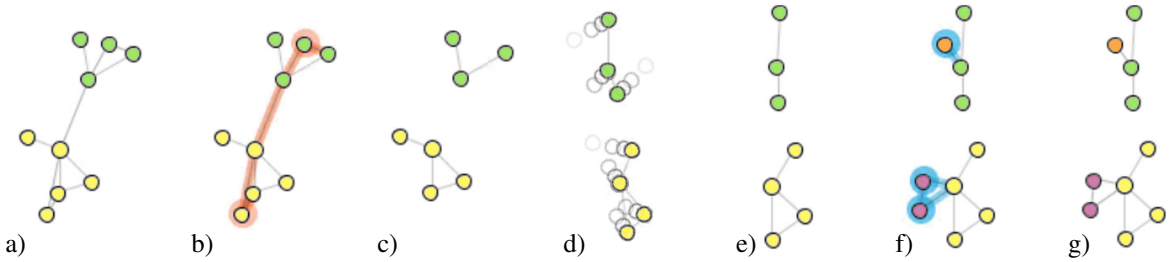


Fig. 1. Staged transitions with change highlighting (node fill colors describe arbitrary, domain-specific, attributes of those nodes): a) initial state, b) element removal (red halos), c) remaining elements only, d) layout adaption, e) remaining elements at their new position, f) element addition (blue halos), and g) final state.

- GraphDiaries, a visual interface designed to improve support for these higher-level tasks and make navigation in dynamic networks more flexible. GraphDiaries features interactive staged animations, non-linear temporal navigation, difference highlighting, small multiples and adapting layout stability. These features can be integrated into any visualization system that supports dynamic node-link diagrams.
- We report on a controlled user study that evaluates the support provided by staged transitions and temporal navigation for tasks related to dynamic network exploration from the above taxonomy. Using representative tasks from our taxonomy, the study compares GraphDiaries to existing time-multiplex interfaces, including: a flip-book, and animated transitions based on linear interpolation.

The remainder of this article is structured as follows: Section 2 presents related work on dynamic network visualization and the use of animations. Section 3 defines our task taxonomy. Section 4 introduces GraphDiaries. Section 5 reports on the controlled user study and discusses its results.

2 Related Work

2.1 Encoding Time in Dynamic Networks

Small multiples are one of the most common representations of dynamic networks. Any static network visualization system can be used to generate snapshots of networks at any time step; these snapshots can then easily be displayed using a tabular view [2][3][4]. Social ego-networks have been visualized by laying out nodes' neighborhood in a radial fashion, each time step corresponding to a radial layer [4]. While this method works well to observe connectivity changes in the neighborhood of an individual node, it does not generalize to the whole network. Overall, small multiples provide an overview of the network's evolution but suffer from the tradeoff between snapshot size on screen and the number of snapshot shown on that screen: while larger images show more details of the network, smaller images mean that more time steps can be shown simultaneously.

A **difference graph** between two [5][6] or more [7] networks makes it possible to directly compare time steps of a network, but difference graphs alone are insufficient to explore long sequences of time steps as typically found in dynamic networks. Color has been used to convey long-term changes, encoding the first appearance of nodes and edges in a **single aggregated image** of the entire network [8]. However, this method makes it hard to encode other temporal measures such as *the time between two connections in a node pair* or particular network-specific attributes such as node type. Values are often aggregated and the goal is to provide a single measure to describe the entire dynamic behavior [9][10], thereby omitting information about the individual time points. A single image enhanced with explicit encoding of temporal changes also increases the visual complexity of the network representation, and important temporal information can get lost if not encoded explicitly.

An alternative to encoding temporal information using color is to use the **third spatial dimension to encode time**: nodes get extruded, becoming thick 3D segments, edges being depicted as bridges between them [11][12]. However, static 2D network visualization is already challenging due to node overlap and link crossing. Adding a third dimension and extra marks to connect vertices across time increases clutter, in addition to the other traditional pitfalls of 3D visualization (visual occlusion, need for extensive and tedious 3D navigation).

Generally speaking, it is always possible to visualize a particular graph metric over time using, e.g, time-series charts. However, this does not work when visualizing the network's evolving topology; questions related to, *e. g.,* connectivity of a sub-graph at a specific time step, or questions that involve many attributes, are almost impossible to answer this way. With GraphDiaries, we aim to provide a consistent visual interface combining the advantages of small multiples – providing an overview that helps users situate the representation in the time dimension – with the explicit encoding of difference images and the flexibility of animated transitions.

2.2 Animation and Temporal Navigation

Animations can be an effective means to decrease complexity by multiplexing the states of a dynamic network in *time* rather than in *space*. Animations as a means to convey changes in user interfaces has been extensively studied in psychology [1], human-computer interaction [13], and information visualization [14]. They have proven useful to switch between statistical information graphics [15], between scatterplot views of multivariate graphs [16], and to highlight changes within textual document histories [14]. Heer *et al.* [15] describe animated transitions in data graphics and report that users generally prefer slower animations. They also recommend to

use *staged transitions* instead of parallel ones, even if they report that "the advantages are not overwhelming". Chevalier *et al.* [14] avoid staged transitions to shorten transition time which, in turn, is an important factor for the efficient use of short-term memory when interpreting animations.

Staged animations have also been used for visualizing dynamic trees [17][18], e.g., when expanding subtrees in SpaceTree [19] or when navigating through the tree's changes over time [20]. DOITrees [21] also make use of animated transitions for the same purpose, but run the different types of animations (subtree expansion, layout adaption) in parallel. In addition, nodes that appear or disappear are briefly highlighted.

Eades and Huang [22] were the first to describe animations to improve the understanding of changes between time steps in dynamic networks. Friedrich and Eades [23][24] describe transitions with several stages for visualizing changes in dynamic networks: first removing network elements, then transforming the entire network so that nodes get as close to their final position as possible, then moving each node individually to its final position, and finally showing new network elements. While added nodes grow or fade-in and removed nodes shrink or fade-out, links are not animated. Although node movements can be tracked, it is difficult to track many changes that happen to the topology, especially for large networks made of unconnected components. Visone [25] features three-stage transitions: first fade out nodes and their incident edges; then remove and add edges between nodes that remain in the network while also updating nodes' positions; and finally add new nodes and their incident edges. We find it difficult to track changes in this type of transition, especially because edges both appear and disappear in two different stages, with the second stage featuring both types of changes simultaneously.

The effect of animations on users' understanding of transitions between two states of the data structure has been the subject of controversy. Robertson *et al.* [26] show that animations are less effective than small multiples and traces for the visualization of trends in scatterplots. Saraiya *et al.*'s study comparing small multiples and animation in networks [27] did not yield conclusive evidence about which technique is more effective at conveying changes. In the area of dynamic graph drawing, studies have found that small multiples is significantly faster than animation on a number of tasks with no statistically significant difference in error rate [28][29]. For questions related to the appearance of nodes and edges, animation has been shown to significantly reduce error rates [28]. Also, difference graphs have been empirically shown to help answer questions about large scale changes in dynamic graphs [30]. Comparing animations and small multiples, Farrugia and Quigley [29] found animations to be less accurate than small multiples. The small multiples conditions contained only four images, and the authors do not blame animations in general, but rather the lack of support for good interaction and navigation.

Comparing the results of all these studies does not lead to a simple answer regarding the usefulness of animations for dynamic networks. Although animations barely increased performance for most tasks, users consistently ranked them high in preference. User feedback also reveals some drawbacks of animations: distraction, longer run time, which in turn increases task time. All these results together reveal that many important aspects of animations are still not well understood; pacing, staging, ordering of stages, graphical rendition of transitions (smooth/abrupt trajectory, fading, zooming, etc.). More experiments, involving higher-level tasks and larger data sets, are required to better understand the challenges and find interaction techniques that efficiently support dynamic network exploration.

In many applications displaying dynamic networks, animations are precomputed, and navigation in time is linear. Systems such as Gephi, TempoVis or Visone only provide a simple time slider [2][31][32][25]. Gephi [31] also features a range-slider that lets users specify a time-span over which to aggregate steps and visualize the corresponding changes in a single frame.

2.3 Layout Strategies for Dynamic Networks

A crucial aspect of dynamic network visualization is how the network is laid out at each time step. Gephi [31] employs an iterative layout solver to provide continuity when interactively changing the displayed time range. While this method provides some continuity between steps, it is not stable: tasks that require revisiting time steps are hard to accomplish, as the layout of a given time step can change depending on the previously visited step. To avoid these changes, TempoVis creates one layout per time step, calculated from the layout of the previous time step [32] and linearly interpolates node position and node color between steps.

While this method can introduce larger changes to the layout over time, Eades argues in favor of a global layout stabilization to better preserve users' *mental map* [33]. The mental map is the image users have of the information and preserving it implies minimizing changes in the visual representation. A globally optimized layout assigns stable positions to all nodes over time, favoring the tracking of elements at the expense of compactness, a more readable layout for each step considered individually, makes it more difficult to track nodes over time. A study by Purchase *et al.* [34] suggests that either global stabilization or local optimization should be preferred, rather than intermediate solutions. Archambault *et al.* [28] compare small multiples and animations under the condition of mental map preservation and suggest that stabilizing the layout does not improve performance. However, in a later study, Archambault and Purchase [35] show that stabilized layouts better support network exploration, a finding confirmed in another study by Ghani *et al.* [36].

These experimental results suggest that a trade-off should be made depending on users' context. A layout stabilized across all time steps will likely not be optimal for any of those steps. A layout optimized for each time step is likely to be very unstable across steps. Decisions about the type of layout, and the type of animation to use, if any, highly depends on the actual tasks users have to perform. Although we consider layout as orthogonal to any of the other features explored in GraphDiaries, the chosen layout strongly influences their appropriate usage, as detailed in Section 4.

2.4 Task Taxonomies

Task taxonomies exist for information visualization in general [37], for static networks [38], as well as for temporal data [39][40]. However, in order to guide the design and evaluation of interfaces for dynamic networks, an effective task framework is necessary that properly reflects the characteristics of data and allows for estimating task complexity.

In their taxonomy about static networks, Lee *et al.* [38] list seven types of network entities: *nodes*, *links*, *paths*, *graphs* (or *sub-graphs*), *connected-components*, *clusters*, and *groups*. These entities are involved in low-level tasks, derived from the taxonomy of visual analytics tasks proposed by Amar *et al.* [37], such as *retrieve value*, *sort*, and *find extremum*.

For dynamic networks, Ahn et al. [41] developed a task taxonomy that includes three dimensions: *graph entities*, *graph properties*, both following the definitions of [38], and *temporal features*. While properties capture attributes that can change over time (degree, centrality, etc.), temporal features describe the type of change that affect these attributes (growth, convergence, stability, etc.). Tasks are specified by first selecting entities, then selecting properties, and finally selecting a temporal feature. The specification process is iterative, and analysts change task components during analysis. While this taxonomy is very detailed and lists many important aspects of temporal changes in dynamic network, we considered it too complex for our purposes and too focused on dynamic network *analysis*. We need a simpler but more systematic taxonomy to better understand the temporal aspect of tasks and how these aspects differ across tasks so as to provide better support for temporal navigation.

3 Task Taxonomy

Our taxonomy is inspired by the static network tasks taxonomy by Lee et al. [38], and combines it with a framework for geo-temporal tasks, proposed by Peuqet [39]. To categorize questions about arbitrary spatio-temporal entities, Peuquet [39] mentions three dimensions: location (*Where* is an object?), time (*When* does something take place?), and object(s) (*What* objects or attributes are observed?). Each task consists of referencing values from two of these dimensions and searching for the answer in the remaining dimension:

- **What + when = where**: describe the location where a specific object is present at a given time;
- **When + where = what**: describe an object that is present at a certain location at a given time;
- **Where + what = when**: describe a time when a certain object is present at a given location.

Peuquet's framework was developed with fixed geographical/spatial locations in mind. In dynamic networks however, there are no fixed spatial positions. Positions of nodes depend on the chosen graph layout, and can vary significantly over time as the graph structure evolves. In Peuquet's framework, moving objects and their attributes are considered as instances of the *What* dimension. In order to keep the framework simple, we need to slightly re-define the *Where* and *What* dimensions.

3.1 Task Dimensions

When: Temporal Tasks—Values for the time dimension include a *particular time* (snapshot), *two times*, a *period*, and *all times*. Besides asking about *when*, we consider attributes such as *how often*, *how fast*, *how long*, *in what order* [42, page 316], *during*, *starts* [43] to name just a few. Examples:

- T_1 *When does node n disappear?*
- T_2 *When are nodes n_1 and n_2 connected?*
- T_3 *How long does it take until clusters c_1 and c_2 merge completely?*

Where: Topological Tasks—With the (spatial) *Where* dimension we refer to nodes, links and their attributes, as well as all higher-level topological structures described in Lee *et al.*'s taxonomy. Peuquet's geographical question *Where on the terrain?* becomes *Where in the graph structure?*; *which node(s)*, *which cluster(s)*, *which links(s)*, *which path(s)*, *which subgraph(s)*, *which motif(s)*, and *which attribute(s)*. Examples:

- E_1 *Which nodes keep the exact same neighbors between time steps t_1 and t_2?*
- E_2 *Which two nodes are connected only once?*
- E_3 *Which cluster is the most unstable over time?*

What: Behavioral Tasks—Our *What* dimension captures the type of change and the behavior of network elements. For instance, nodes and links can *appear* or *disappear*, are *present* or *absent*, clusters can *grow* or *shrink*, *merge* or *split*. Examples:

- B_1 *How does the degree of node n evolve over time?*
- B_2 *What happens to cluster c between t_1 and t_2?*
- B_3 *Are nodes n_1 and n_2 connected at time t?*

3.2 Compound and Higher level Tasks

We are now able to describe *compound tasks* as combinations of low-level tasks. For example, *"How big is group g when node n leaves it?"* can be split into three low-level tasks: 1) where is n (what=presence of n, when=all steps)? 2) when does n leave g (where=n, what=leaves g)? and 3) what is the size of g at time step t (where=g, when=t)? The more low-level tasks, dimensions and values in these dimensions are involved, the more complex and high-level a task is.

Very high-level tasks consist of complex operations such as *describe trends, anomalies and change behavior, compare changes, find outliers and correlation,* and *analyze dependencies between changes*. These tasks require human judgement and strategies to decompose them into low-level tasks. Such very high-level tasks are not explicitly described in the current taxonomy. They have to be decomposed into simpler tasks covered by the taxonomy.

3.3 Translating Graph Tasks to Visualization Tasks

Visualization systems for dynamic networks should provide effective visual representations and interactions to support the tasks described in this taxonomy, and possibly others related to specific application domains. The translation of graph-oriented tasks into perception tasks and interactions is not straightforward: it requires some degree of familiarity and experience with the visual representation.

In this article, we focus on the node-link visual representation because it is the most popular. This representation encodes graph topology using points for vertices, lines for edges, and relies on a layout algorithm to place the nodes on the plane. Graph attributes are visualized using the visual attributes associated with the points and lines. All the tasks on dynamic graphs described in this section need to be translated into a series of perception tasks and interactions, and this translation requires some learning, leading to some acquired knowledge on the properties of the visual representation.

In particular, trained users know that, in addition to reading lines to understand the topology of a graph, connected nodes are closer-by than unconnected ones, with the exception of "bridge" nodes connecting two distant groups and that can be detected with their long connecting lines. This is a property enforced by most layout algorithms, although it is sometimes relaxed for small graphs where lines are easy to read, allowing nodes to be evenly spaced. Otherwise, this property is essential for larger graphs where lines should be drawn lightly to limit occlusion and cannot be read easily. Therefore, topology tasks can be performed by either reading the lines, if possible, or looking at the proximity of nodes. For larger graphs, densely connected nodes become clouds where bridges can be noticed with their long outstanding lines.

Most of the studies on graph readability have focused on relatively small graphs where participants had to read the lines to perform topology tasks. In this article, we are interested in higher-level tasks which involve tracking groups of nodes and estimate the type and quantity of change. From a visual perspective, these tasks require both proximity inspection and reading lines when necessary and possible.

While we do not claim that our taxonomy is comprehensive — the number of possible tasks being virtually infinite as acknowledged in [38] — we believe it is useful, as it allows us to describe the components of any interface or system in terms of what dimensions (and combinations thereof) they cover; and thus, what low-level and compound tasks they support. The following high-level implications for the design of dynamic network visualization systems can be drawn from it:

When—Specific time steps must be easy to identify and reach, so that users can compare and analyze them in detail. Features that can meet these requirements include overviews of the network's time steps, mechanisms for the quick selection and filtering of those steps, and a flexible scheme to navigate from one step to another.

Where—Elements with particular properties must be easy to identify and situate in the network's topology, and to track along time steps. The layout and rendering of the network should be parameterized carefuly.

What—Understanding the nature of, and possibly quantifying, the changes that graph elements undergo requires that the corresponding attributes be easy to identify. Those should be emphasized. w These dimensions also allow us to discuss the complexity of tasks, based on which dimensions are required and involved to solve a given task. This taxonomy also helped us structure and operationalize the experiments reported on later in this paper, informing the selection of representative tasks that cover all three dimensions.

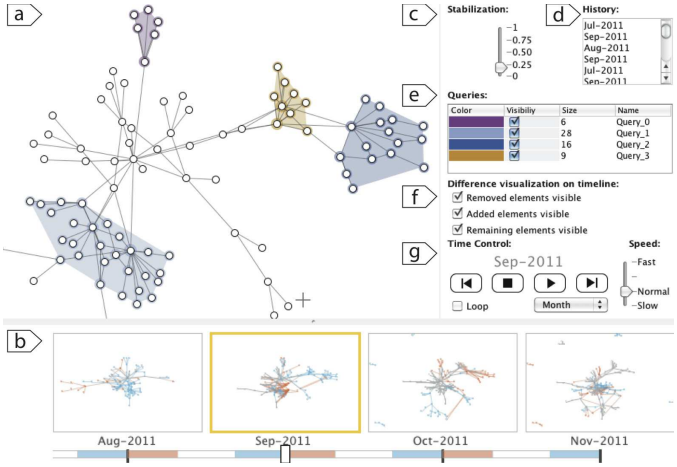


Fig. 2. GraphDiaries interface: a) Network view, b) Timeline, c) Layout stabilization slider, d) Navigation history, e) Node queries, f) Panel to change visibility of red, blue or gray elements in the Timeline, g) Animation playback panel.

4 GraphDiaries

GraphDiaries is designed to help users answer questions related to the different dimensions of the above taxonomy: *Where*, *What* and *When*. GraphDiaries relies on an *interactive staged animated transition* technique that highlights changes from one time step to another, as described below. The main network view (Figure 2-a) shows the network as a node-link diagram at the time step currently selected in the Timeline (Figure 2-b). The network view focuses on answering questions about the *Where* and *What* dimensions.

The *When* dimension is the primary focus of the thumbnails and the slider in the Timeline. Each thumbnail shows the network at a particular time step. *Difference highlighting* shows differences between any given step and the previous one, using the same visual encoding as in our staged transitions (Section 4.2): removed elements are colored red, new ones blue, remaining ones gray.

Additional interface components, shown in Figure 2, provide further support to relate the taxonomy's three dimensions. The layout slider (c) controls layout stability (*where*), as explained in Section 4.1. Navigation history (d) and time control panel (g) provide high-level playback and access to time steps (*when*) in the network view (*where*). Users also have control on the temporal granularity of the timeline (hours, days, weeks..). Conversely, options in (f) let users configure what information is shown (*what* type of changes) in the timeline's thumbnails (*when*). Finally, the node query panel (e) lists node queries created by users, as detailed in Section 4.5.

4.1 Dynamic Graph Layout

Layout stabilization (Figure 2-c) enables users to choose between a globally optimized layout for all time steps, one

locally optimized for each time step, or any configuration in-between obtained by linear interpolation. Although the idea of a slider for partial stabilization is not new [3], our implementation is actually independent of the underlying graph layout. We were interested in how differently stable layouts combine with the different transitions to support users. Indeed, depending on the layout strategy, the same transition allows to track different types of changes; combined with change highlighting, a stable layout draws users' attention to regions of the network that are changing (based on the amount of red and blue in different regions of the representation). While for an optimized layout, transitions support tracking changes in entities' positions, thus reflecting their new neighborhood (Section 4.3).

Global and locally-optimized layouts are computed as follows: the global layout is computed with LinLog on the whole time-aggregated graph, taking into account the number of edges between node pairs as edge weight. The locally-optimized layouts are computed for all time steps t_i, by running the faster Fruchterman-Reingold layout algorithm [44] on each time-step independently. For force-based algorithms that perform iterative improvements, instability is avoided by starting from an initial layout obtained by interpolating between the layout at the previous time step t_{i-1} and the global layout. Nodes that appear at this point are initially positioned at their coordinates in the global layout. All layouts, global and local, are stored in memory, so that the exact same layout can be reused when navigating back to the same time step. As the user manipulates the slider, the interpolated layout is calculated and then relaxed to remove overlaps. We found an almost fully locally-optimized layout (80% local layout) to be a good tradeoff for the various tested data sets, and which we use as default value.

4.2 Staged Animated Transitions

Staged animated transitions with change highlighting are designed to help users understand *what* changes occur in the network's topology while navigating through time steps.

4.2.1 Design Goals

In addition to common design goals for animations such as smoothness, aesthetics and intuitiveness [23], [14], the design of our transitions is based on the following criteria:

- D1 **Separation of Concerns**—Staged transitions avoid overloading users with too much information. Staging allows them to focus on each type of event in turn. Low-level changes, which account for all types of higher level changes, can be split in three stages: *node and edge removal*, *node repositioning* and *node and edge addition*.
- D2 **Visualization independence**—The visual encoding of transitions must not interfere with the visual encoding of network data (node and edge shapes, color, visual elements etc.) or any user driven annotations (selection of nodes) so as to be applicable to all kinds of existing node-link visualizations.
- D3 **Controllability**—Users should be able to control the animation speed, freely navigate inside the frames of a transition, and interrupt animations at will. Controllability is important for two reasons: it enables users to focus on and understand complex changes, possibly playing them back and forth multiple times at low speed; it also enables them to quickly browse through or skip transitions or particular stages within transitions of low interest to them.
- D4 **Ad-hoc Transitions**—As the user interacts with the visualization, he or she should be able to change the graph layout and the timeline's granularity. Any transition should be calculated on demand, taking into account the current layout and visible graph elements. This includes the ability to show transitions between non-adjacent time steps to allow comparison of arbitrary time steps.

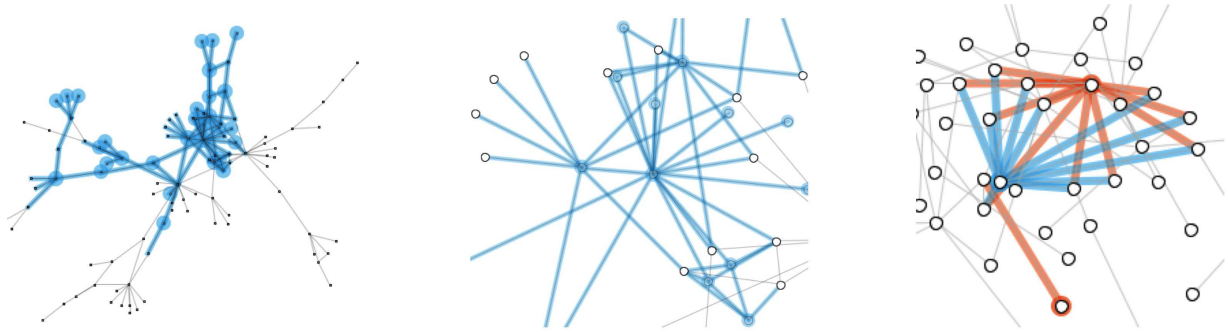
We explored various implementations of the above goals, iterating on the interaction design and fine-tuning the parameters through pilot tests. We compared the different options considered for both the staging of transitions and the interactions that control temporal navigation. The following sections describe our final design, relating the features of the original design goals.

4.2.2 Transition Stages

Staged transitions in GraphDiaries can be triggered between any two time-steps t_i and t_j, not necessarily adjacent (D4). Stages correspond to the three types of low-level topological changes (D1), as illustrated in Figure 1.

1) **Remove Elements** (300 ms)—A red halo fades-in around each node and edge that is no longer present in t_j (Figure 1-b). Edge halos fade-in slightly later than node halos to emphasize the perception of affected nodes in clusters and dense regions. Then, all elements involved in the removal fade-out along with the associated halos.
2) **Transform Layout** (600 ms)—Remaining nodes and connections get smoothly moved to their new position in the layout of t_j using a slow-in/slow-out pacing function (Figures 1-c, d, and e). This stage has a longer duration to help users track node position changes. Changes to domain-specific attributes, encoded using, e.g., node fill color or node size, also get animated during this stage.
3) **Add Elements** (300 ms)—This last stage adds new nodes and edges by fading them in, accompanied by blue halos that vanish thereafter (Figure 1-f).

We tested multiple alternative designs for our transitions. We discarded the option of first showing element insertion, then changing to a new layout, and finally showing element removal, because it significantly increases the number of visual elements on screen during the transition. These elements must potentially be moved in the second stage, causing additional distraction. Furthermore elements which are present in both time steps are not distinguishable. We also considered separating node and edge changes in two stages, but this increases transition time even more by making the transition appear less smooth and changing regions harder to identify. Furthermore, it introduces ambiguities such as the following: a node disappearing implies that its incident edges disappear as well; but when first removing edges and then nodes, users whose attention gets caught by a particular node might no longer know whether this node was connected or isolated



(a) Overview with changes (b) Detail with changes (c) Direct difference

Fig. 3. Size of node halos is independent from the zoom level, allowing for analysis at different levels of scale. (a) A high distance emphasizes changing subgraphs, while b) a close distance reveals details. Figure c) shows the direct difference between two time steps and reveals the replacement of a central node.

before the staged transition started. We also tried to run stages in parallel, either partially or completely, *e. g.,* repositioning nodes while removing or adding elements, or start fading-in new elements before the fade-out of elements removed ended. However, we found that this option contradicts D1 in the sense that users cannot focus on a particular type of change; staging allows them to anticipate when a certain type of change will happen. Overall, we observed that as networks become denser and as changes are more frequent, staging should be favored over shorter-but-more-confusing transitions that run all animations in parallel (fade-in of new nodes, fade-out of removed nodes, repositioning of remaining nodes).

4.3 Change Highlighting

We use halos to highlight changing nodes and edges rather than coloring them directly, so as to avoid interfering with existing visual encodings (D2), instead making it possible to visually encode, for example, temporal network measurements such as dynamic centrality, or domain-specific data attributes [9][10]. Figure 1 shows that halos are still visible when node fill color encodes a domain-specific data attribute. A recent study by Archambault et al. [30] confirms that using color to highlight changes between two graphs increases users' performance, compared to a simple animation or no animation. Further evidence about the benefits of explicit change highlighting in comparing diagrams is found in Zamen *et al.*'s work [45]. While there is no strong agreement in the community about which colors to use to encode those changes, we argue that red and blue are relevant choices as they feature a significant contrast in hue and are readable by people impaired by color blindness.

Halos around nodes and edges have a constant, scale-independent thickness, which guarantees that changes will always be clearly visible, no matter the network's size and zoom level (Figure 3(a-b)). Holding the *shift*-key while hovering any small multiple in the timeline view highlights the direct differences between the network in the Network view (Figure 3-c) and the one in the hovered thumbnail; again, blue elements are present only in the hovered thumbnail, while red elements are only present in the current (reference) time step.

4.4 Temporal Navigation

The duration of each stage of a transition was fine-tuned manually. While a total duration of 1.2 seconds might seem long, it is necessary to actually enable users to keep track of the complex changes that occur (D1). However, as users might not always be interested in all stages of a transition depending on the task at hand (depending on the *what* component of the task), we enrich the animated transitions with interaction techniques that let users quickly skip or fast-forward them (D3), while navigating through time (*when*).

We defined methods to navigate over time and interactively control staged transitions both across time steps (inter time-step navigation) and within transitions (intra time-step navigation). In the timeline, red, white and blue sections visually identify the three stages of a transition, thus facilitating intra time-step navigation when dragging the slider (Figure 2-b).

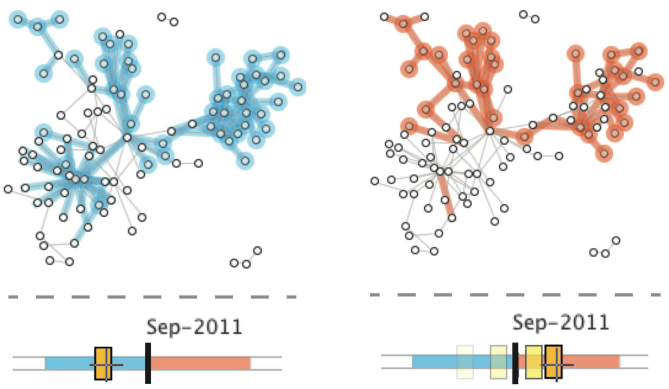
4.4.1 *Inter Time-Step Navigation*

Users move between adjacent time steps using the left and right arrow keys. To jump between non-adjacent time steps without going through the intermediate ones, users simply click on the corresponding thumbnail. In both cases, the main graph view gets smoothly animated according to the staged transition technique described earlier, providing details about *what* happens *where* in between the two time steps.

4.4.2 *Intra Time-Step Navigation*

Users can control a staged transition's unfolding in various ways, depending on whether they are trying to get an overview of changes through time (*what, when*), are tracking a particular element over time (*where, when*), or searching for a particular event (*when*): We provide four options for controlling the duration of and the position withing a single transition so that users can adapt navigation to the current task, either overview, tracking or searching for a particular event related to a graph entity (*what*) or a location (*where*):

a) **Run to completion**—Pressing and holding down the left mouse button on a thumbnail, or keeping an arrow key depressed, runs the full staged transition.
b) **Interrupt and finish**—Releasing the mouse button or key while the transition is running interrupts it. The remaining



(a) Additions in September 2011 (b) Removals in October 2011

Fig. 4. Dragging the yellow time cursor in the timeline around *September 2011* shows what nodes and edges were added during Sep. 2011 (blue elements) and what nodes and edges were removed in Oct. 2011 (red elements). The example shows that a major part of new nodes added in Sep. have been removed in Oct.

stages are played fast-forward (200 ms), all in parallel, in order to guarantee perceptual continuity and help users preserve their mental map of the network.

c) **Skip animations**—Clicking on a thumbnail or hitting an arrow key (quick press/release) jumps to the target time step without any animation. Users can browse through steps very fast. This is useful when the details of changes between two particular steps are not so important.

d) **Interact**—Controlling the animation's pace with a time slider can be very effective when exploring transitions. We support this through direct manipulation of the time slider, with the red, white and blue zones between steps delimiting the three stages of the transition (Figure 4).

The Time Control Panel mentioned earlier (Figure 2-g) provides standard playback controls, including playback speed, looping and temporal aggregation. Changing temporal granularity creates temporal aggregation of the network's steps. Thumbnails get updated and the graph view shows the same transitions as described in Section 4.2.

4.5 Dynamic Node Queries

Changing layout and disappearing nodes make it particularly hard to keep track of specific sets of nodes and subgraphs. GraphDiaries provides a mechanism to highlight node sets over time. This feature, called *node queries*, is similar to the selection highlighting feature available in ScatterDice [46]. Queries are created by lasso selection and are represented by a colored halo around the nodes that are part of the query, plus a convex hull polygon that encompasses all those nodes. Figure 5 shows a node query during an animated transition. Queries can be refined by the user at any time to include new nodes. Node queries are managed in the user interface panel depicted in Figure 2-e.

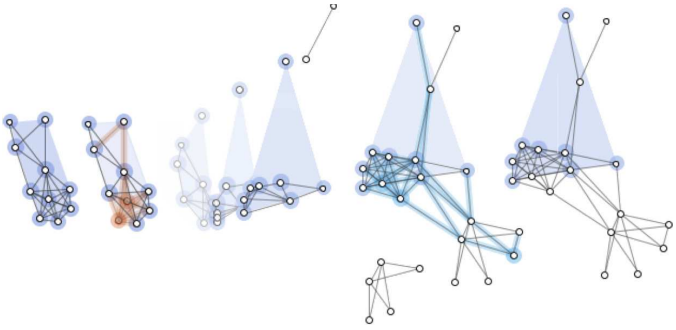


Fig. 5. Steps of a transition featuring a node query.

5 User Study

To evaluate the potential benefits of staged transitions and associated interactive navigation techniques, such as the possibility to smoothly navigate between non-adjacent time steps, we conducted two controlled experiments. The first experiment measured participants' performance on a set of three tasks, each one covering one of the dimensions of our taxonomy (*Where, What, When*). The second experiment was a follow-up study to obtain additional empirical data about participants' ability to assess instability in dynamic networks.

Compared to previous evaluations of graph visualization techniques, we favored higher-level tasks that involve observation, tracking and comparison of attribute based subgraphs, not necessarily corresponding to a single connected component. Our tasks possibly require non-linear navigation patterns, for example, revisitation of a given time step multiple times, compare non-adjacent time steps, and tracking different kinds of changes between steps. For graph navigation, we believe low-level tasks, such as tracking one specific node over time or detecting the presence of particular edges in two specific time steps, do not properly reflect realistic network exploration tasks and would not be very informative as to the techniques' efficiency. However, tasks that are too high-level by requiring extended graph knowledge or experience in mapping user tasks into visual tasks, are hard to control and to compare in a controlled user study.

5.1 Techniques

The primary goal of this study was to assess the potential benefits of techniques that support navigation in time and the impact of different visual feedback strategies to convey changes. Evaluating all factors and combinations of techniques in GraphDiaries is beyond the scope of a single paper and would require multiple user studies, each one looking at a subset of factors in isolation. For a first assessment of our design choices, we compared GraphDiaries to two baseline techniques: video animation and flip book. We compared conditions in which different time navigation capabilities were enabled. The interface components made available were the same across conditions: only the graph view (Figure 2-a), and the timeline with small multiples (Figure 2-b). The three conditions were as follows:

FB: **Flip Book** provided a static representation of the graph at each time-step, like an image viewer or a file explorer

with content preview. Users could switch between any two images but there were no animation between time steps. Graphs were replaced instantaneously. To jump between time steps participants either clicked on the thumbnail in the timeline, or used arrow keys.

VA: **Video Animation** allowed participants to navigate using a video player metaphor, as in [32][31]. Animations were shown between consecutive time steps only and showed all changes at the same time: added nodes and edges fading-in, removed ones fading-out, and all others moving to their position in the target step's layout. Animation time was 1 second. Participants could play back and pause, as when watching videos. As in the FB condition, participants could either click on the corresponding thumbnail in the timeline or use arrow keys to navigate between consecutive steps.

GD: **Graph Diaries** provided participants with the major techniques presented earlier in this paper, extending the capabilities of VA: *staged animation* with *change highlighting* between any step, *inter and intra time step navigation*, and *difference visualization* on the thumbnails. Because of the staged nature of the transitions, default animation time was set to 1.2 seconds, as explained earlier. All other features of GraphDiaries: layout stabilization, node queries and history, were removed.

All techniques used the layout strategy described in Section 4 with an stabilization of 20% (almost fully locally-optimized layout). Node positions were calculated and stored once for each data sample in order to ensure the exact same layout across techniques. Labels where shown on-demand when hovering nodes. All datasets fully fitted on screen at nominal scale. Panning and zooming were thus disabled to avoid noise in the experimental data due to uncontrolled differences in participants' spatial navigation strategies.

5.2 Tasks

Participants were asked to answer questions about a real-world co-authorship network made of more than 10,000 authors from 200 research groups between 2005 and 2009 (5 time steps). Nodes represent individual authors. In order to convey group membership, authors that belonged to the same research group shared the same color. For each task, we used data samples consisting of approximately 100 nodes in 7 groups (average, per time step). Group membership was not directly related to the network's topology. In a given time step, an edge links two nodes when the two authors have collaborated on at least one publication during the corresponding year. Participants had to answer the following questions:

T_{size} **— *(When)* In which year is the red group largest?** Participants had to navigate through all time steps and compare them, in order to find the time when the red group was largest. To input their answer, they had to press the space bar and select the correct year from a pop-up menu.

T_{inst} **— *(Where)* Which group features the most changes over time?** Groups exchanged nodes over time, *i. e.,* they lost some nodes and gained new ones. Participants had to observe all groups over the years, and eventually click any node from the group that featured the most changes.

T_{trend} **— *(What)* What is the trend of the red group? Does it grow, does it shrink, does it remain stable, or is it unstable?** Participants had to spot a trend over the years. Groups that grew actually doubled in size. Those that shrank halved. Both did so in a non-monotonic way. Stable groups kept a set of constant core members that was larger than the average size of the group. Unstable groups did not feature any such set of stable core members. They could possibly gain, loose or exchange all of their members over time. To input their answer, participants had to press the space bar and select the correct answer from a pop-up menu.

5.3 Datasets

A major problem with these tasks is that difficulty can vary significantly with each dataset's complexity. Observed differences between techniques can actually stem from this variability if dataset complexity is not controlled and counterbalanced properly across conditions. But comparing the complexity of different datasets is difficult, especially if they have not been created artificially, carefully controlling their characteristics.

To guarantee equivalent conditions, we extracted and analyzed data samples (subgraphs) from our main dataset for each task, and reused them across all participants. Each task required special conditions and tuning to make sure that there would not be too much ambiguity with respect to what answer was the correct one.

We re-used the same datasets *across techniques*, thus allowing for a fair comparison between conditions. Each data sample was used in 3 trials, one per technique. However, simply reusing datasets across conditions would have been risky, as participants might have remembered answers or partial answers from previous trials, resulting in an uncontrolled learning effect. To minimize possible asymmetrical transfer between conditions, we mirrored and rotated the three instances of each network. In addition, each time a dataset appeared, node labels were anonymized by randomly assigning them popular English names at runtime.

The data samples were obtained as follows and were rendered as illustrated in Figure 6. Nodes were filled with the color of the corresponding research group, using the Set1 9-color scheme from colorbrewer2.org, so that participants could identify groups pre-attentively.

T_{size}—Per dataset, seven research groups were randomly extracted from the original network and one group was chosen to be the target one (colored red). The size of that group was analyzed over time and data samples were selected by hand in order to remove sources of ambiguity. A group, at the time step when it was at its largest size, always featured at least two more nodes than in any other time step.

T_{inst}—All 200 groups from the original network were analyzed, based on particular types of changes: *size*, *nodeGain* (nodes added per time step), *nodeLoss* (nodes removed per time step), and number of *constantNodes*. Each dataset was composed of 6 groups with a low rate of change:
$avg(nodeGain) < 3$, $avg(nodeLoss) < 3$, $avg(size) > 6$.
The additional target group was extracted from the original network and featured a high rate of change:
$avg(nodeGain) > 7$, $avg(nodeLoss) > 7$, $avg(size) < 15$.

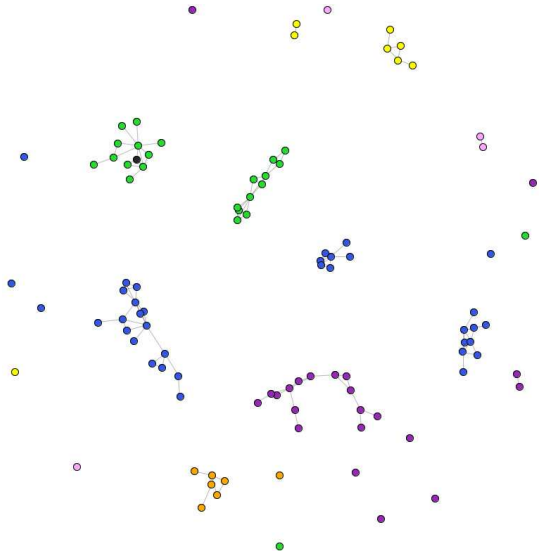


Fig. 6. An example data set of the data as used and laid out in the user study. In the experiment. Nodes were colored by research group. The actual background in the experiment was a very dark gray.

T_{trend}—For each dataset, 7 groups were randomly extracted from the original network. One group was chosen to be the target one (colored red). Datasets were selected by hand to ensure that the target group featured a clear trend, either growing, shrinking, stable, or unstable.

5.4 Design and Apparatus

The first experiment followed a within-subject full-factorial design with the 3 earlier-mentioned techniques ($Tech \in \{$GD, VA, FB$\}$) and the 3 tasks described above ($Task \in \{T_{size}, T_{inst}, T_{trend}\}$) as independent variables. The resulting 9 conditions were counterbalanced using a Latin square, blocking by *Tech*. For each condition, participants were presented with 4 training trials, followed by 5 actual measurement trials. Each trial used a different dataset. The presentation order of datasets was the same for all participants.

On average, the experiment lasted 70 minutes. It was divided into two sessions to avoid fatigue due to the high cognitive load involved in performing the tasks. Two techniques were tested in the first session (50 minutes). The remaining one was tested in the second session (20 minutes), which had to take place at least one hour after the first one. Participants were allowed to rest between each trial.

Participants were instructed to favor accuracy over speed, *i. e.,* to avoid making mistakes. Due to the complexity of the tasks, each trial was limited to 90 seconds. After 60 seconds, the screen flashed briefly, and a countdown for the remaining 30 seconds was shown. Once an answer had been selected, both the right answer and the participant's answer were shown.

We asked eighteen volunteers (four female), ranging in age from 24 to 44 years old to participate in this experiment. All of them used computers daily, had normal or corrected-to-normal vision and were not color blind. The experiment was conducted using a 2.66Ghz iCore 7 MacBook Pro with 4GB of RAM and a monitor resolution of 1440×900. The interface was implemented in Java 6 using the ZVTM toolkit [47]. Background was set to black to minimize visual fatigue. Participants interacted using a mouse and an external keyboard. During training, animation speed decreased from 2 seconds initially to the default duration of 1 second for VA and 1.2 seconds for GD.

The two main measures were error rate and task completion time. The timer started as soon as the dataset showed up on screen, and stopped when participants either clicked a node (for T_{inst}) or hit the space bar (for the other two tasks). Error rate was computed differently for each task. In T_{size}, time slices were ranked according to the size of the target (red) group. For T_{inst}, groups were ranked according to their rate of change: $avg(nodeGain)+avg(nodeLoss)$. Again, error was equal to the position in this ranking. For T_{trend}, the answer was either right or wrong as answers were nominal.

5.5 Hypotheses

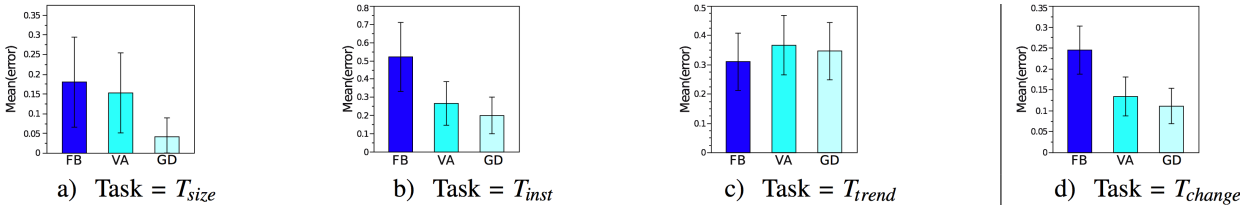
Our hypotheses were as follows:

- H_1 For all three tasks, error rate is lower when using staged transitions (GD), as this technique helps better keep track of changes between time steps.
- H_2 For the same reason, completion time does not increase significantly when using staged transitions (GD), despite the longer duration of animations and their higher visual complexity.
- H_3 Participants use features that enable them to transition between non-adjacent time steps, when available.

5.6 Results

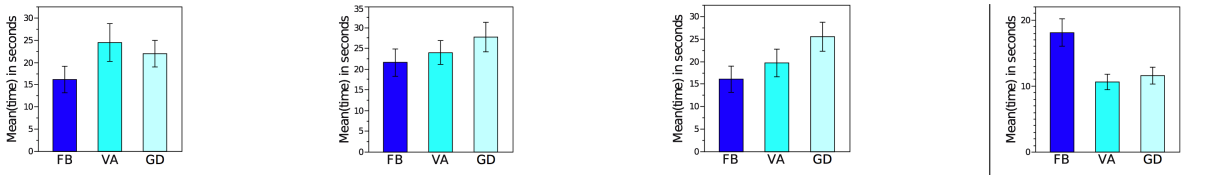
A SHAPIRO-WILK test showed that measurements of time and error were not normally distributed. The measurements distribution could not be corrected using either a logarithmic or BOX-COX transformation. We thus performed a non-parametric MANN-WHITNEY-WILCOXON (Mann-Whitney U) test for pair-wise comparison between techniques, for each task. During the experiment, we realized that one particular dataset used for T_{size} contained changes that were too hard to detect. The difference between the largest two sizes was only 2 nodes, for an average group size of 18 nodes. We subsequently removed the corresponding trials from our analysis, as this noisy set would not have yielded meaningful results.

All analyses are performed by *Task*, as error rates are measured differently across tasks. T_{size} (Figure 7-a): GD features a significantly ($p<0.04$) lower error rate (avg. 4%) than FB (avg. 18%). VA performed similar to FB (avg. 15%) with a near-significant difference compared to GD ($p<0.069$). There was no significant difference between FB and VA. T_{inst} (Figure 7-b): GD features a significantly ($p<0.011$) lower error rate (avg. 21%) than FB (avg. 52%), with VA in-between (avg. 27%) and a near-significant difference ($p<0.072$) compared to FB. There was no significant difference between VA and GD. T_{trend} (Figure 7-c): we did not observe any effect of *Tech* on *Error*. All three techniques feature relatively similar error rates.



a) Task = T_{size} b) Task = T_{inst} c) Task = T_{trend} d) Task = T_{change}

Fig. 7. *Error* rate per *Tech* × *Task*. Error bars show the 95% confidence limit of the mean.



a) Task = T_{size} b) Task = T_{inst} c) Task = T_{trend} d) Task = T_{change}

Fig. 8. *Time* (seconds) per *Tech* × *Task*. Error bars show the 95% confidence interval of the mean.

T_{size} (Figure 8-a): FB (avg. 16.2s) is significantly faster than both VA and GD (avg. 24.5s, p<0.0001 and 21.9s, p<0.0001, respectively). GD was not significantly faster than VA. T_{inst} (Figure 8-b): FB features a significantly (p<0.005) lower time (avg. 21.6) than GD (avg. 28.4s), with VA in-between (avg. 23.9s) and not significantly different from the other two. T_{trend} (Figure 8-c): FB features a significantly lower time (avg. 16.1s) than GD (avg. 26s, p<0.001) and VA (avg. 19.7s, p<0.007).GD is significantly slower (p<0.004) than VA.

5.7 Follow-up Experiment

T_{trend} did not yield significant results in terms of error rate, which is our main measure of performance. Despite intense piloting, such high-level and complex tasks are difficult to control. We decided to redesign this task and run a follow-up study. We switched from four possible answers to two, and created the target group artificially, inserting ground truth in real-world data so as to better control this group's behavior. The new task was as follows:

T_{change}**—(What) How does the red clique behave? Is there a high turnover, *i. e.,* are there many nodes coming and leaving, over time? Or is there a larger constant core of members over all years?** The core was made of nodes that remained in the group from the beginning to the end. If the amount of core nodes was larger than the average size of the group, the group was stable; otherwise it was unstable. Participants had to press the *space bar* and select the correct answer from a pop-up menu.

We extracted datasets from the co-authorship network we had used in the first experiment. For each dataset, 6 groups of nodes were randomly created, and one group was artificially created according to predefined figures to guarantee its stability or instability (depending on the trial). This target group had between 13 and 17 nodes, including a constant core of nodes. Core nodes never left the group. Other nodes remained in the group for one to three years. In the *unstable group* condition, the target group's core size was lower than 50% of the overall group size. In the *stable group* condition, the core's size was above 50%. The difference in size between stable and unstable was either 30%, 15% or 7.5%, corresponding to three levels of difficulty. Although the last value seems fairly small, we did make sure that there was no ambiguity when performing the task visually. It is also important to bear in mind that the task was about observing changes of nodes, rather than changes in group size.

We asked the same 18 volunteers to participate in this experiment. Time between the first experiment and this follow-up study never exceeded two weeks. The only task tested was T_{change}. We compared the same three techniques. We added a new feature to the environment that enabled participants to toggle visibility of all node labels at once. While this can potentially introduce noise in the data, participants needed a way to compare graphs. We kept track of the status of node labels (shown or hidden) in the experiment's logs.

Results for this task show that *Tech* has a significant effect on *Error*. FB features a significantly (p<0.004) higher error rate (avg. 25%) than VA and GD (avg. 13% and 11%, respectively), which are not significantly different from one another (Figure 7-d). *Tech* also has a significant effect on *Time*. As shown in Figure 8-d, FB features a significantly (p<0.001) higher task completion time (avg. 18.1s) than VA and GD (avg. 11.6s and 10.6s, respectively), which are not significantly different from one another.

5.8 User Strategies

Participants were asked to report their strategies and subjective preferences in a post-hoc questionnaire. Further information was retrieved by analyzing low-level interaction logs: keyboard and mouse events, as well as somewhat higher-level information such as the order in which participants visited the five time steps for each trial, when were node labels displayed, how much was the time slider used. Examining these logs, we identified various common and alternative strategies for exploring dynamic networks. The main observation is that temporal navigation in the data follows Shneiderman's mantra [48]: users first wanted to have an overview of the data through time, and only then did they focus on particular time steps in more detail. While this observation holds for all tasks and techniques, other observations were made for specific tasks, as detailed below. We also observed that 27%

of the participants never used the mouse. This means that they never used the slider, and never compared non-adjacent time steps (indeed, the mouse was necessary for both).

T_{size}—Find Time Step—While participants were able to quickly go from time step to time step with FB and GD, things were less straightforward with VA. They either watched the animation step by step, or dragged the time slider. Participants made extensive use of the non-adjacent time step comparison feature with GD, while they barely used it in conjunction with FB. While this feature was not available in VA, it is noteworthy that participants did not employ any feature for comparison between time steps with VA: in most cases, they simply navigated over time using video controls.

T_{inst}—Find Network Entity—This task was about spotting nodes entering and leaving the group over time. In addition to the amount of color that was shown with groups, the middle step of staged transitions (GD) made it easy to identify how many (and which) nodes did not change. If a group was much smaller during this stage than at the beginning or end, it was probably unstable. However, changes had to be observed throughout the network. Participants employed very different strategies; FB allowed for quick navigation which led them to quickly visit all time steps in linear order, mostly using the keyboard. With VA, participants iterated less frequently over all time steps, except those who used the slider, the latter making navigation faster. 50% of all participants who used the mouse compared non-adjacent time steps.

T_{trend}—Characterize change—This task was the most difficult, and we did not observe any statistical difference between conditions. However, we could observe that participants applied a strategy similar to that employed in T_{inst}; they observed all time steps first and heavily switched between steps in the GD condition.

T_{change}—Characterize change—This task was the only one where enabling and disabling all node labels at once was allowed. Since this task was about tracking nodes joining and leaving a group, we expected the task to be very hard in the FB condition. However, despite the difficulty of the task, results indicate that very few participants (four) actually used labels in FB and in VA. Among them, all but one also displayed labels in the GD condition. This seems to indicate that having to read and memorize node labels was considered as inconvenient and unnecessary. As expected, many participants (80%) switched between non-adjacent time steps.

5.9 Discussion

The goal of this study was to investigate the potential benefits of staged transitions and other features such as the possibility to smoothly navigate between non-adjacent time steps and to control the type and speed of transitions. We wanted to assess their impact on user comprehension of the high-level changes that occur in dynamic networks. Our results show that animations significantly decrease error rate, and suggest that staged transitions offer further improvements for some tasks, thus partially supporting H_1. Animations increase task completion time for tasks T_{size}, T_{inst}, T_{trend}, but decrease it for T_{change}. Comparing the first three tasks, the difference is significant in terms of statistical analysis, but its magnitude is relatively small. Thus, H_2 is supported. A lower error rate should be favored over a small decrease in task completion time in real work situations, and we interpret those results as generally favorable for GD. Quantitative results are in accordance with user preferences collected through the post-hoc questionnaire: GD was ranked as the best techniques by 80% of the participants, and FB was ranked last by 72% of them. Our results are in accordance with Archambault *et al.* [28] who found that for tasks related to node and edge appearance, animations decrease the number of errors when compared to small multiples, but increase task completion time. Further studies [45][15] suggest that users prefer accurate techniques that they can trust to faster, but less reliable, ones.

Error—Results suggest that animations played a major role in participants' performance: the two techniques that relied on it, VA and GD, yielded consistently and significantly lower error rates than FB (Figure 7). Animations seemed to play a crucial role when tracking incoming and outgoing nodes. When animations were not available, labels sometimes helped to solve the task, but significantly increased cognitive load and fatigue according to several participants. When animations were available, the simple fading of nodes and edges provided by VA helped, but the animation was often not sufficient to help participants track more than a few changes in the limited screen area. Changes scattered throughout the screen were hard to track, as suggested by the results of T_{inst}, in which the entire network had to be observed. GD provided an advantage in that respect, as it used color to convey changes in a pre-attentive manner and enabled participants to spot and identify the kind of changes in a much larger area. Subjective feedback indicates that participants found highlighting *overall useful* (44%) or *useful* (36%). 66% of them indicated that it made them more confident about their answers.

The significantly lower error rate of GD for T_{size} suggests that highlighting of changes (halos) played an important role when comparing two time steps. Indeed, solving this task required comparing local maxima between non-adjacent time steps; this was made relatively straightforward by both FB and GD, but highlighting was only enabled in GD.

Time—Task completion time was slightly lower with GD than with VA for T_{size}, and slightly higher for all other tasks (Figure 8). We tentatively attribute this difference to the fact that FB and GD enabled direct comparison of arbitrary time steps. FB was always the fastest technique, except for task T_{change}. This is probably due to the fact that participants had to read and memorize labels with FB, for lack of a more efficient and less cognitively-demanding alternative. It is interesting to note that the average time spent solving a task with FB was far below the maximum time per trial (90 seconds). Participants seldom reached that limit, even if they were not particularly confident about their answer; they considered that spending more time on the task with FB was not worth it, as it was not going to provide more insight. Overall, FB featured lower task completion times mainly because participants gave up, not because they were confident about their answer, as confirmed by the higher error rates with this technique and our observation of participants' behavior during the experiment. This suggests

that providing users with animated transitions while enabling them to skip these transitions is a good solution.

Strategies—Only five participants never used the mouse for interacting with the system; all others adopted non-linear time-navigation patterns in the FB and GD conditions. The most typical pattern was to get an overview by quickly looking at all time steps, then focus on a particular subset of—not necessarily contiguous—steps, switching back and forth to compare them. This supports H_3 and suggests that features enabling direct difference visualization between arbitrary time steps is useful, as those spare users from having to rely on their memory to compare distant time steps, thus lowering the associated cognitive load.

Tasks—The experimental tasks were chosen according to the dimensions of our taxonomy. However, we had to make some compromises to make those tasks amenable to a controlled experiment. Designing a study and operationalizing such high-level and cognitively-demanding tasks is very challenging, as acknowledged in [28]: we had to pilot and iterate on the design, datasets and tasks multiple times. This necessarily entails some limitations. Our tasks have mainly focused on the appearance and disappearance of nodes as opposed to links. As mentioned earlier, some topological tasks involving group membership turn into visual tasks of looking for geometrical proximity: nodes close to one another are more likely to be connected, especially when using a locally-optimized layout (see Figure 6). Tracking groups essentially comes down to tracking node movements *and* node appearance/disappearance. While staging separates both types of change for better perception, change highlighting results in a higher or lower concentration of red or blue halos in different regions of the representation. Further studies, however, should also include links, for instance asking questions about density and connected components; questions which require some domain expertise to answer, putting some constraints on the participant population.

One other limitation is the number of time steps associated with each dataset. Real-world data will often feature more than five steps. Future studies should investigate the effect of this specific factor, but it already seems reasonable to claim that some features discussed earlier, such as the capability to directly compare and smoothly transition between non-adjacent time steps, should bring even stronger benefits in situations where the number of time steps is larger. Another limitation of this study is that we only considered networks made of approximately 100 nodes. However, very large networks and the more complex tasks associated with them require aggregation and analytical capabilities that are both complementary to the work presented here and beyond the scope of the generic navigation tasks considered.

Finally, we did not fully isolate all factors. Our goal was to compare the transition and navigation techniques as complementary elements, not to isolate the effects and contributions of each single low-level interaction technique (this would require many more conditions). We thus cannot say much about the respective contribution of, *e. g.,* small multiples and animation to performance improvements. Our results in terms of comparison between plain and staged animations are in accordance with those of Heer and Robertson's study about transitions in data graphics [15]. Archambault *et al.*'s study [35][36] suggests that layout stabilization depends very much on the task, and that small multiples and animations are rather complementary for different tasks. Results from these studies and ours help gain a broader perspective, but further studies are required to gain a more comprehensive understanding of the various factors that come into play.

6 Summary and Future Work

Staged animated transitions are designed to support the temporal exploration of dynamic networks. They help users identify changes between time steps by highlighting them. Coupled with flexible interaction techniques that let users bypass animations or accelerate their playback smoothly, they can significantly improve task performance. We evaluated GraphDiaries against techniques commonly found in visualization systems for temporal graph navigation. We observed a minor increase in task completion time, that is compensated by a significant decrease in error rate in favor of animated transitions. The latter improve the perception of changes and provide users with a rich set of exploration strategies.

The design of our techniques was informed by a taxonomy of tasks related to the exploration of dynamic networks. We chose to classify tasks based on their categorization along three dimensions: *time*, *type of change* and *graph entities*. This taxonomy can help guide the design of user interfaces other than GraphDiaries that provide support for temporal exploration.

Future work should explore the issue of how to convey the temporal evolution of node and edge attributes: we want to extend the scope of GraphDiaries towards the exploration of evolving multivariate and hierarchical graphs, whose complexity will require new aggregation, navigation, comparison and tracking techniques.

References

[1] B. Tversky, J. B. Morrison, and M. Betrancourt, "Animation: Can it Facilitate?" *Human-Comp. Stud.*, vol. 57, no. 4, pp. 247–262, 2002.

[2] M. Pohl, F. Reitz, and P. Birke, "As time goes by: integrated visualization and analysis of dynamic networks," in *Proc. AVI*. ACM, 2008, pp. 372–375.

[3] P. Federico, W. Aigner, S. Miksch, F. Windhager, and L. Zenk, "A visual analytics approach to dynamic social networks," in *Proc. i-KNOW*. ACM, 2011, pp. 1–8.

[4] M. Farrugia, N. Hurley, and Q. Aaron, "Exploring temporal ego networks using small multiples and tree-ring layouts," in *Proc. ACHI 2011*, 2011, pp. 23–28.

[5] K. Andrews, M. Wohlfahrt, and G. Wurzinger, "Visual graph comparison," in *Proc. IV*. IEEE, 2009, pp. 62–67.

[6] B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J.-D. Fekete, "Weighted graph comparison techniques xfor brain connectivity analysis," in *Proc. CHI*, 2013, pp. 483–492.

[7] M. Hascoët and P. Dragicevic, "Interactive graph matching and visual comparison of graphs and clustered graphs," in *Proc. AVI*. ACM, 2012, pp. 522–529.

[8] C. Collberg, S. Kobourov, J. Nagra, J. Pitts, and K. Wampler, "A system for graph-based visualization of the evolution of software," in *Proc. SoftVis*. ACM, 2003, pp. 77–86.

[9] K. Lerman, R. Ghosh, and J. H. Kang, "Centrality metric for dynamic networks," in *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, ser. MLG '10. New York, NY, USA: ACM, 2010, pp. 70–77.

[10] P. Federico, J. Pfeffer, W. Aigner, S. Miksch, and L. Zenk, "Visual analysis of dynamic networks using change centrality," in *Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on*, 2012, pp. 179–183.

[11] T. Dwyer and P. Eades, "Visualising a fund manager flow graph with columns and worms," in *Proc. IV*, 2002, pp. 147–152.

[12] U. Brandes and S. R. Corman, "Visual unrolling of network evolution and the analysis of dynamic discourse," *Information Visualization*, vol. 2, no. 1, pp. 40–50, 2003.

[13] M. Shanmugasundaram and P. Irani, "The effect of animated transitions in zooming interfaces," in *Proc. AVI*. ACM, 2008, pp. 396–399.

[14] F. Chevalier, P. Dragicevic, A. Bezerianos, and J.-D. Fekete, "Using text animated transitions to support navigation in document histories," in *Proc. CHI*. ACM, 2010.

[15] J. Heer and G. Robertson, "Animated transitions in statistical data graphics," *TVCG*, vol. 13, pp. 1240–1247, 2007.

[16] A. Bezerianos, F. Chevalier, P. Dragicevic, N. Elmqvist, and J.-D. Fekete, "Graphdice: A system for exploring multivariate social networks." *Comput. Graph. Forum*, pp. 863–872, 2010.

[17] K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst, "Animated exploration of dynamic graphs with radial layout," in *Proc. InfoVis*, 2001, pp. 43–50.

[18] D. Guilmaine, C. Viau, and M. J. McGuffin, "Hierarchically Animated Transitions in Visualizations of Tree Structures," in *Proc. AVI*. ACM, 2012, pp. 514–521.

[19] C. Plaisant, J. Grosjean, and B. Bederson, "SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation," in *Proc. InfoVis*, 2002, pp. 57–64.

[20] S. K. Card, B. Sun, B. Pendleton, J. Heer, and J. Bodnar, "Time Tree: Exploring Time Changing Hierarchies," in *Proc. VAST*. IEEE, 2006, pp. 3–10.

[21] J. Heer and S. K. Card, "DOITrees Revisited: Scalable, Space-Constrained Visualization of Hierarchical Data," in *Proc. AVI*, 2004, pp. 421–424.

[22] P. Eades and M. L. Huang, "Navigating clustered graphs using force-directed methods," *Graph Algorithms and Applications*, vol. 4, no. 3, pp. 157–181, 2000.

[23] C. Friedrich and P. Eades, "The Marey Graph Animation Tool Demo," in *Proc. Symposium on Graph Drawing*. Springer, 2001, pp. 95–107.

[24] ——, "Graph Drawing in Motion," *Graph Algorithms and Applications*, vol. 6, no. 3, pp. 353–370, 2002.

[25] U. Brandes and D. Wagner, "Visone – Analysis and Visualization of Social Networks," in *Graph Drawing Software*. Springer-Verlag, 2003, pp. 321–340.

[26] 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, Nov. 2008.

[27] P. Saraiya, C. North, and K. Duca, "Comparing Benchmark Task and Insight Evaluation Methods on Timeseries Graph Visualizations," in *Proc. BELIV*. ACM, 2010, pp. 55–62.

[28] D. Archambault, H. Purchase, and B. Pinaud, "Animation, Small Multiples, and the Effect of Mental Map Preservation in Dynamic Graphs," *IEEE TVCG*, vol. 17, no. 4, pp. 539–552, 2011.

[29] M. Farrugia and A. Quigley, "Effective temporal graph layout: A comporative stydy of animations versus statuc display methods," *Information Visualization*, pp. 47–64, 2011.

[30] D. Archambault, H. C. Purchase, and B. Pinaud, "Difference map readability for dynamic graphs," in *Proc. of Graph Drawing*, ser. LNCS, vol. 6502, 2011, pp. 50–61.

[31] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *Proc. ICWSM*, 2009.

[32] J. Ahn, M. Taieb-Maimon, A. Sopan, C. Plaisant, and B. Shneiderman, "Temporal Visualization of Social Network Dynamics: Prototypes for Nation of Neighbors," in *Proc. Social comp., Behav.-Cult. mModeling*, 2011, pp. 309–316.

[33] B. B. Bederson and A. Boltman, "Does animation help users build mental maps of spatial information?" in *Proc. InfoVis*, 1999, pp. 28–35.

[34] H. C. Purchase and A. Samra, "Extremes are better: Investigating mental map preservation in dynamic graphs," in *Proc. Diagrams*. Springer-Verlag, 2008, pp. 60–73.

[35] D. Archambault and H. C. Purchase, "Mental map preservation helps user orientation in dynamic graphs," in *Proc. of Graph Drawing*, ser. LNCS, vol. 7704, 2013, pp. 475–486.

[36] S. Ghani, N. Elmqvist, and J. S. Yi, "Perception of animated node-link diagrams for dynamic graphs," *Computer Graphics Forum*, vol. 31, no. 3pt3, pp. 1205–1214, 2012.

[37] R. Amar, J. Eagan, and J. Stasko, "Low-Level Components of Analytic Activity in Information Visualization," in *Proc. InfoVis*. IEEE, 2005, pp. 111–117.

[38] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry, "Task taxonomy for graph visualization," in *Proc. BELIV*. ACM, 2006, pp. 1–5.

[39] D. J. Peuquet, "Its about time - a conceptual-framework for the representation of temporal dynamics in geographic information-systems," *Annals of the Association of American Geographers*, vol. 84, no. 3, pp. 441–461, 1994.

[40] N. Andrienko and G. Andrienko, *Exploratory Analysis of Spatial and Temporal Data*. Springer, 2006.

[41] J. Ahn, C. Plaisant, and B. Shneiderman, "A Task Taxonomy for Network Evolution Analysis," HCIL, University of Maryland, Tech. Rep. HCIL-2012-13, 2012.

[42] A. M. MacEachren, *How Maps Work*. Guilford Press, 1995.

[43] J. Allen, "Towards a general theory of action and time." *Artificla Intelligence*, pp. 123–154, 1984.

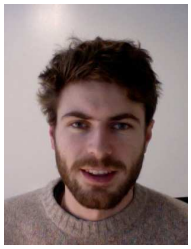
[44] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Softw. Pract. Exper.*, vol. 21, no. 11, pp. 1129–1164, Nov. 1991.

[45] L. Zaman, A. Kalra, and W. Stuerzlinger, "The effect of animation, dual view, difference layers, and relative re-layout in hierarchical diagram differencing," in *Proc. of Graphics Interface*, 2011, pp. 183–190.

[46] N. Elmqvist, P. Dragicevic, and J.-D. Fekete, "Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation," in *Proc. InfoVis 2008*, 2008.

[47] E. Pietriga, "A toolkit for addressing hci issues in visual language environments," in *Proc. VL/HCC*. IEEE, 2005, pp. 145–152.

[48] B. Shneiderman, "The eyes have it: a task by data type taxonomy for information visualisation," in *Proc. VL*. IEEE, 1996, pp. 336–343.



Benjamin Bach is currently a PhD student in the Aviz Group at INRIA in Paris, France. He received his Masters Degree in 2010 from the Department of Computer Science at the University of Technology, Dresden, Germany. His research interests include network visualization, visualization of temporal and multidimensional data, and search interfaces.

Emmanuel Pietriga received a PhD degree in computer science from INPG (France) in 2002. He worked for INRIA and Xerox Research Centre Europe, did his postdoctoral research at MIT as a team member of the W3C. He is now working for INRIA in France and Chile as a research scientist. His research interests include interaction techniques for multi-scale user interfaces, wall-sized displays, and visualization techniques for massive datasets.

Jean-Daniel Fekete received his PhD in Computer Science in 1996 from Université Paris-Sud. He joined the Human-Computer Interaction Laboratory at the University of Maryland in the USA for one year. He was recruited by INRIA in 2002 as a confirmed researcher and became Senior Research Scientist in 2006. He is the Scientific Leader of the INRIA Project Team AVIZ that he founded in 2007. His main Research areas are Visual Analytics, Information Visualization and Human Computer Interaction.