DiffAni: Visualizing Dynamic Graphs with a Hybrid of Difference Maps and Animation

Sébastien Rufiange and Michael J. McGuffin

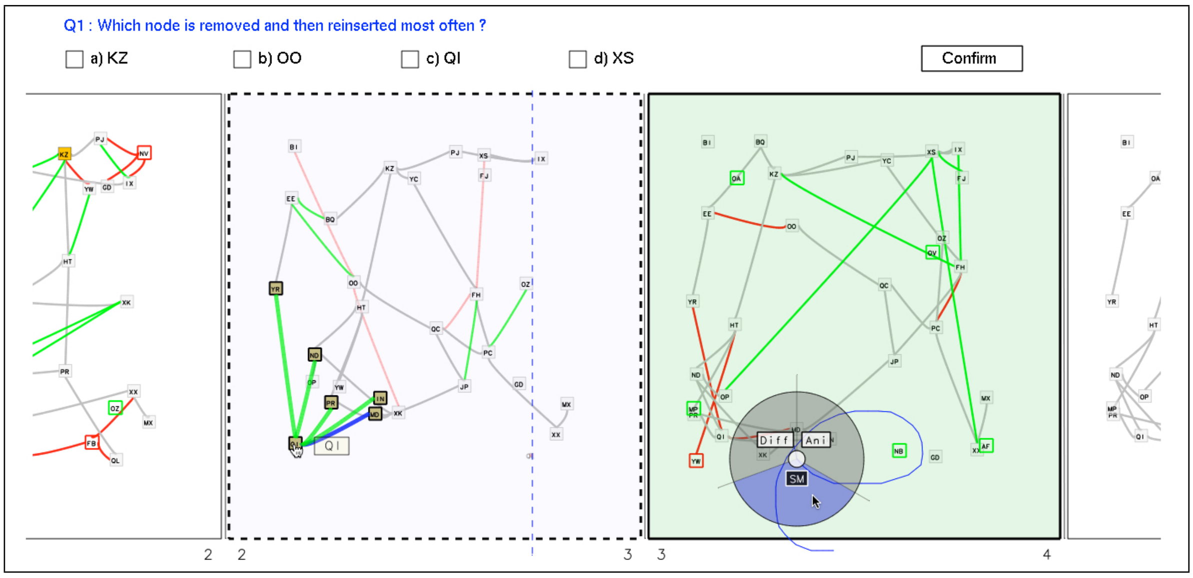

Fig. 1. Given a dynamic graph defined over a series of time slices, our DiffAni hybrid visualization displays it as a sequence of consecutive tiles. Each tile may show one or more time slices, and there are three kinds of tiles: diff tiles (indicated with a solid border and two time slice numbers below them, e.g., the left-most tile), animation tiles (indicated with a dashed border and two time slice numbers below them, e.g., the tile covering time slices 2 to 3), and small multiple tiles (indicated with a solid border and a single time slice number below them, e.g., the right-most tile). Nodes and edges are colored in red if they are being removed, or green if they are being added, over time slices. The radial menu above is being used to convert a diff tile into two small multiple ("SM") tiles.

Abstract—Visualization of dynamically changing networks (graphs) is a significant challenge for researchers. Previous work has experimentally compared animation, small multiples, and other techniques, and found trade-offs between these. One potential way to avoid such trade-offs is to combine previous techniques in a hybrid visualization. We present two taxonomies of visualizations of dynamic graphs: one of non-hybrid techniques, and one of hybrid techniques. We also describe a prototype, called DiffAni, that allows a graph to be visualized as a sequence of three kinds of tiles: diff tiles that show difference maps over some time interval, animation tiles that show the evolution of the graph over some time interval, and small multiple tiles that show the graph state at an individual time slice. This sequence of tiles is ordered by time and covers all time slices in the data. An experimental evaluation of DiffAni shows that our hybrid approach has advantages over non-hybrid techniques in certain cases.

Index Terms—Dynamic networks, hybrid visualization, taxonomy, evolution, animation, difference map

1 Introduction

Within the field of graph visualization [38], a significant frontier for research is dealing with time-dependent graph data. Many real-world networks change over time, including social networks, communication networks, migration of people between cities or countries, international trade networks, and network models of the relationships between source code modules. The visualization of such dynamic graphs is challenging for at least two reasons. First, in computing the layout of a graph for each moment in time (each time slice), there is a trade-off between optimizing the layout quality for that particular time slice, versus reducing the movement of nodes across time slices to improve the users' *mental map*. So far, empirical evaluations (e.g., [28, 29, 1]) have not yielded simple, clear conclusions about the effect or optimal level of mental map preservation. A second challenge is that there are several ways the time slices of a graph can be visually presented, including small multiples, animation, and 3D representations. Here again, there are trade-offs, for example: when compared to animation, small multiples require more space (or they require the user to sacrifice spatial resolution to fit all of the small time slices on a single screen), however small multiples have the advantage that the user can compare different time slices with fast eye movements rather than waiting for an animation to complete or replay. In addition, as with the first challenge mentioned, previous empirical comparisons [1, 11, 42] of different representation methods have yielded mixed results. It seems likely that the relative advantage of each visual representation depends on several factors, including the size and density of the graph, the degree

- *Sébastien Rufiange is with École de technologie supérieure, Montréal, Canada. E-mail: sebastien@rufiange.com.*
- *Michael J. McGuffin is with École de technologie supérieure, Montréal, Canada. E-mail: michael.mcguffin@etsmtl.ca.*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

of mental map preservation, and the task being performed by the user. For example, if the user is interested in the total number of nodes in a graph as it changes over time, then tracking individual node movements is not necessary, and a small multiples representation may be best. However, if the user must track the movements of a particular node, without the benefit of highlighting, then smooth animation may prove superior.

Previous studies seem to support this possibility, with some [1, 11] finding that static representations are superior, and another [42] finding that animation is better. A reasonable user interface, therefore, might allow users to view a dynamic graph either as small multiples or as an animation, or even show both in side-by-side coordinated views. This would present other problems, however. Different subsets of the dynamic graph data might best be visualized with different representations. Showing only one representation at a time in a single view could require the user to frequently switch between representations, as well as cause disorientation during switching. On the other hand, using multiple coordinated views would consume more screen space.

We therefore propose mixing representations together in a novel hybrid visualization of dynamic graphs. We have implemented this idea in a prototype called DiffAni (pronounced to rhyme with Tiffany). DiffAni displays the network as a horizontal sequence of *tiles*, where each tile is either (1) a static small multiple showing a single time slice, (2) a static difference map that is colored to show differences between two time slices (we call these diff tiles), or (3) an animation clip that smoothly interpolates between two time slices. These three kinds of tiles can be intermixed, but always show time slices in their chronological order, from left to right, with the left-most tile corresponding to the beginning of the dataset, and the right-most corresponding to the end. Such a hybrid visualization gives the user the flexibility to change the representation over any time interval, and has the potential to display each temporal portion of the network with the best representation for it, without consuming the additional screen space that would be required with a multiple coordinated views approach.

The rest of this paper presents our contributions, which are (1) a taxonomy of non-hybrid strategies for visualizing dynamic graphs; (2) a taxonomy of hybrid visualizations involving focal and context regions; (3) our prototype DiffAni that allows users to select any consecutive set of time slices and change their representation, and also allows users to scroll across multiple "diff" frames and navigate within multiple animation frames all with a single, unbroken mouse drag gesture; and (4) the results of a controlled experiment that show that DiffAni can sometimes yield performance superior to that with a non-hybrid visualization.

2 Related Work

2.1 Visualization of Dynamic Graphs

Visualization of time-varying data, in general, is of growing concern, as evidenced by a recent taxonomy [10]. In the specific case of graphs, a dynamic network [8, 34] can evolve in different ways over time. Most previous approaches for visualizing dynamic graphs can be classified according to our own taxonomy in Figure 2, which shows each node with a changing color representing a numerical attribute. Other kinds of network changes, such as topological changes, are of course also possible, and will be the focus of section 3 and the later sections of this paper. Small multiples (Figure 2.1) show snapshots side-by-side, and can be thought of as nesting 2D layouts of the graph within an external time axis. The opposite approach is to nest the time axis within the elements of a 2D layout (Figure 2.3). The graph's time axis can also be mapped to *user* time, resulting in an animation (Figure 2.2), or to a 3rd spatial axis, resulting in a 3D visualization (Figure 2.5). Finally, if the graph is linearized along a single axis, as with an arc diagram [3, 39], the 2nd spatial axis can be used for time (Figure 2.4). Variants of these approaches can also be created using adjacency matrices rather than node-link diagrams.

Small multiples and animation (Figures 2.1 and 2.2) have been the most common strategies in previous work [28, 1, 11, 42]. Examples of nesting the time axis inside the network layout (Figure 2.3) include [35] (which uses a node-link representation for the graph) and [41, 5]

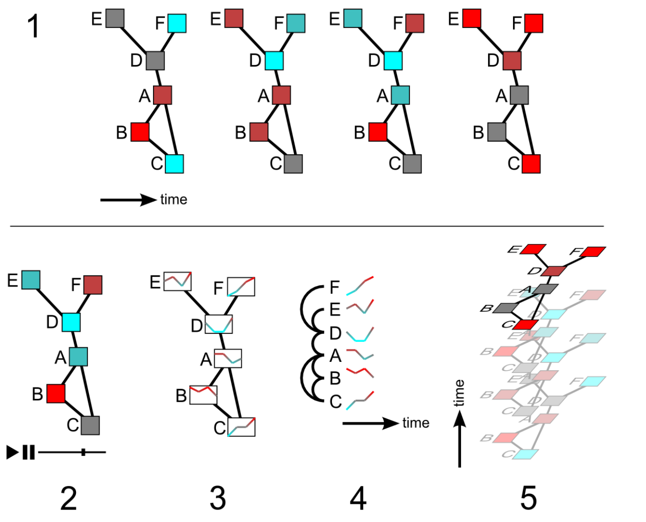

Fig. 2. A taxonomy of strategies for visualizing dynamic graphs. Colors indicate a changing numerical attribute associated with each node. (For simplicity, the above figure shows changes in node attributes, however most of the above approaches could be adapted to instead show changes in topology, which are the kinds of changes discussed in the rest of this paper.) **1**: small multiples. **2**: animation. **3**: sparklines or other glyphs embedded within the graph's layout. **4**: the time axis is perpendicular to an arc diagram. **5**: a 3D visualization, with time along the 3rd spatial axis.

(which use matrices). Having nodes laid out along one direction and time along another direction (Figure 2.4) is used in [36, 16]. A variant of this idea is used in [32, 30], where nodes are first clustered, then clusters are laid out along one direction, and time along another, to show the merging and splitting of clusters over time. Finally, the 3D visualization (Figure 2.5) was used in [4, 14]. Burch et al. [6, 7] have also proposed very original, but somewhat complicated, ways to visualize dynamic graphs that don't fit within Figure 2.

Note that three of the strategies (Figures 2.1, 2.2 and 2.5) can optionally use some kind of color or shape coding to indicate which nodes or edges are different in each snapshot, compared to the previous and/or next snapshots. *Difference highlighting* is a design dimension that is orthogonal to the possibilities in Figure 2, and can be combined with small multiples, animation, or 3D. This has been called a "difference map" [2] or "difference layer" [42], and has been shown to be beneficial [2].

In section 3 and onward, we will use the terms "small multiple" when no difference highlighting is used, "diff" for a static difference map that uses coloring to highlight differences between two (not necessarily consecutive) time slices, and "animation" for animations that also use difference highlighting. Our prototype supports all three of these representations. The reasons we will focus on these three representations are (1) they are the most studied approaches to date, compared to the others in Figure 2; (2) they are more scalable than arc diagram (Figure 2.4) or matrix-based approaches, which both require the height of the visualization to grow linearly with the number of nodes; (3) matrix-based approaches make it harder to find paths [15]; (4) the chosen techniques allow all kinds of changes to a network to be depicted, whereas the approaches in Figures 2.3, 2.4 have difficulty showing topological changes or changes in node position; and (5) they avoid the occlusion and navigation problems of 3D (Figure 2.5).

2.2 Comparison of Animation and Small Multiples

Previous studies of visualizations that use animation have obtained mixed results. For example, [37] argue that animation is often misused or less effective than static representations. However, there are also examples of visualizations that were found to benefit from animation (e.g., [17, 19, 9]), though these did not involve dynamic graphs.

Three previous papers [1, 11, 42] have experimentally compared small multiples and animations for dynamic graphs. In two of these [1, 11], small multiples were generally found to be superior to animation in terms of time to complete tasks. However, in both studies, when tasks required the user to examine specific nodes (e.g., "Which node has a degree that remains constant?"), either the nodes to examine were highlighted with unique colors across all time slices [1], or else all nodes in the graph were displayed with a mix of colors and shapes [11]. This makes it easier for the user to identify the node(s) of interest in each time slice, by simply looking for the appropriate color (or color and shape combination). A more general or realistic situation could have all nodes displayed with the same color and shape, in which case we could expect animation to yield more benefit when nodes are changing location from one time slice to the next. In partial support of this, Zaman et al. [42] displayed nodes with the same color and shape across time slices. In their 2nd experiment, where nodes changed location, they found animation was superior to (static) "difference layers".

We also note that animation usually involves playback of a sequence of images, by pressing a key or a "play" button, at a speed determined before the animation starts. When the user, or more often the programmer, chooses the playback speed, there is a trade-off between how easy the animation is to understand, and how quickly one can view the complete sequence. We suspect that a more useful way to view an animation is by continuously dragging the pointing device (mouse), so that the user can continuously control the speed, and stop and reverse at any moment to review events of interest. This kind of interaction is often possible by dragging along a time slider widget. However, all three studies may be biased against animation here. In [1, 11], the time slider widget was small and therefore, by Fitts' law [26], time consuming to acquire with the mouse, and in [42] there was no time slider at all. In real-world software, where users only casually view videos, navigation options are usually limited to a play/pause button and a small time slider. However, professional video editing software sometimes allow users to drag anywhere within a window to navigate within time, e.g., by using a special mouse button combination.

The experimental evaluation we present later in this paper is designed with realistic, expert use in mind. Therefore, in our work, nodes are not highlighted in a way that eases their identification across time slices, temporal navigation in animation is always done by dragging, and it can be performed almost anywhere in the main view.

2.3 Hybrid Visualizations

Previous work have explored the possibility of combining techniques to visualize a network at one moment in time [21, 33]. A paper [18] also explored the usefulness of hybrids in the context of dynamic networks by nesting representations inside a different one. For instance, the structure of a graph is first depicted with a node-link diagram, and then other parts of the network are shown using different representations (e.g., complexity plot, matrix). However, this technique cannot be used to split a timeline into several parts, to try to benefit from using varying representations at different times in an evolving graph. Another work [20] performed a user study to compare different node duplication techniques that could be used with the NodeTrix hybrid [21] to improve the understanding of social networks, but did not involve dynamic graphs.

There are different ways to combine representations in information visualization in general, and these possibilities were discussed in [24]. While their classification is not focused on network visualizations, our work includes a juxtaposition mechanism, which is similarly used in multiple coordinated views (e.g., [31, 12, 40, 42]). In contrast, the kind of hybrid approach we propose can reduce the need to switch between several views (e.g., difference view, animated view) by merging them and allows splitting a history into smaller parts. Multiple coordinated views can show several representations at once, but do not allow the more flexible and interactive mixing of visualizations as hybrids, and consume more screen space or make each representation smaller.

3 Taxonomy of Hybrid Visualizations with Focal and Context Regions

A graph can evolve over time and different visual techniques can be used to show these changes (such as small multiples, diff and animation). An interesting direction of research (also mentioned in [1]) is to explore how these approaches can be mixed together. Imagine that we can interactively select different representations for each transition of a dynamic graph, depending on which one might be more appropriate in some context. Are the new combinations beneficial, and in which cases? We use a taxonomy (shown in Figure 3) to organize possible combinations of visualization techniques for dynamic graphs.

Taxonomies can help explore the design space of potentially useful combinations and can be constructed in different ways. A first step is to verify which techniques might be worth combining together. For example, node-link representations can help follow paths and are more

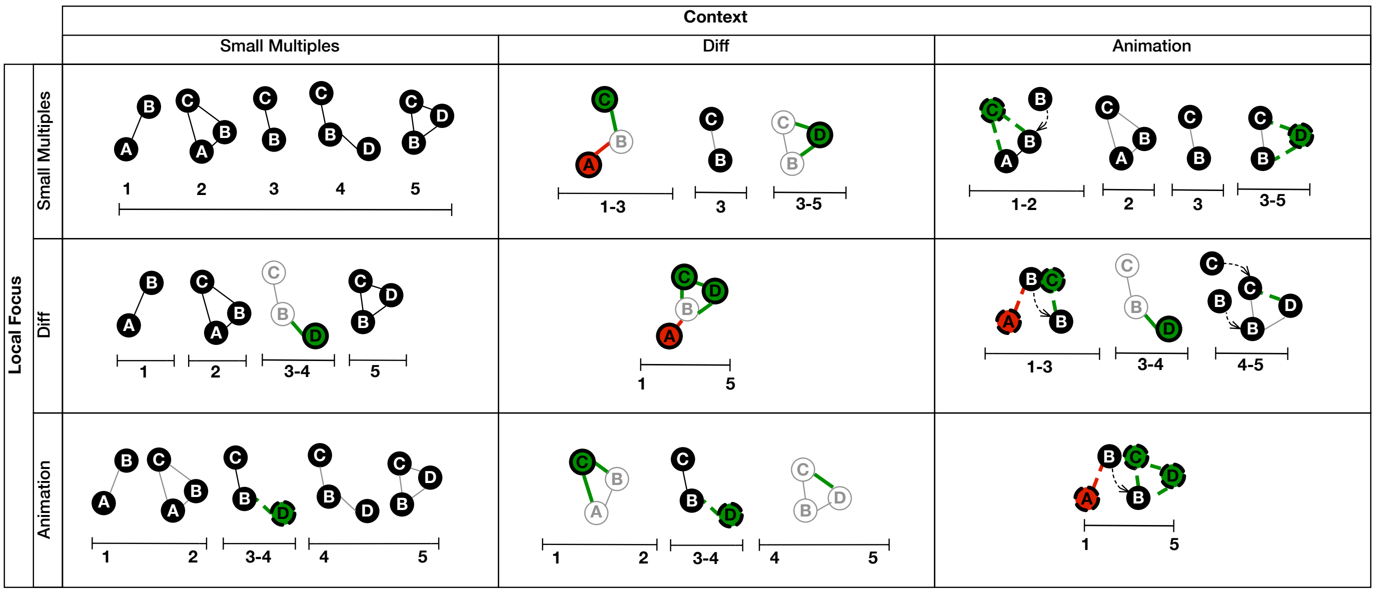

Fig. 3. A taxonomy of different hybrid visualizations of the same dynamic graph. The context is the visualization technique used for surrounding time slices. Other time slices (i.e., the local focuses) can also be visualized using different techniques. The cells along the diagonal show "pure" (non-hybrid) techniques, whereas other cells show possible hybrids. Dashed lines are used to illustrate that some of the changes can be animated.

familiar to users than matrices. However, matrices can be more scalable, even for dense networks. Since they are complementary in some ways, combining them may be useful (as shown in TreeMatrix [33] and NodeTrix [21]). Elastic Hierarchies [43] also aims to combine advantages of node-link diagrams and treemaps for visualizing trees. Similar reasoning lead us to combine visualizations to help understand dynamic graphs, since animation should help tracking moving nodes, whereas diff could enable quick identification of topological changes. We focused on investigating combinations that seemed more useful in practice (i.e., where the possible advantages of the combination were more clear).

In the taxonomy, we illustrate how different visualization techniques for dynamic graphs (i.e., small multiples, diff and animation) can be mixed in various ways. A small multiple represents a single "photo" of a dynamic network, taken at a specific moment in time. A difference map allow to highlight the changes between two small multiples, by combining them. Animations gradually interpolates changes made to a network, e.g., by moving and fading elements in and out. The combinations of these visualizations are expressed in terms of Focus+Context (as in [43, 33]) based on the idea that a main technique (the context) is used for a graph, but for some reason (e.g., possibly to optimize the screen space or better see movements), the user wants to use a different representation for a subgraph (the focus).

The timelines of a dynamic graph are shown in each cell of the taxonomy, along with possible visualizations for specific time steps and transitions. An evolving graph can thus be split in various ways and a different visual technique can be used for a region in "focus". For instance, in the middle column of the top row of the taxonomy, diffs are generally used to represent the dynamic graph, except at time step 3, where a small multiple is used instead. The opposite cell (in the left-most column of the middle row) illustrates another possible combination, i.e., insertion of a diff to highlight the changes between time steps 3 and 4, among several small multiples.

Another case (in the right-most column of the middle row) shows how a diff could be used instead of an animation in the middle transition of an evolving network. This could be useful if, for example, nodes do not move a lot for some period of time, and thus using an animation might not be the best approach in this case. In our prototype implementation, the user can interactively swap visualizations according to his/her own preferences (or heuristics), and for any time slices. Thus, it covers all the possibilities illustrated in our taxonomy, as well as similar ones, e.g., using two different focuses (diff and animation) in a context of small multiples.

4 Prototype

A video overview of our DiffAni prototype is available online[1]. Three kinds of tiles are supported: *small multiple* tiles (with no difference highlighting), *diff tiles* highlighting differences between two (not necessarily consecutive) time slices, and *animation tiles* (that also use difference highlighting). In difference highlighting, nodes and edges are shown in red or green if they are removed or added, respectively, across time slices. Currently, topological changes are the only kinds of changes visualized by the prototype. In addition, when the user place the mouse cursor over a node in a tile, the edges incident on that node, as well as the node's neighbors, are highlighted.

The user may use the mouse to zoom and pan over the sequence of tiles. The user can also drag sideways to navigate in time within animation tiles. Inside such a tile, the positions of nodes are interpolated based on the current position in time, and nodes and edges that appear or disappear are gradually faded in or out. We decided to use this scrubbing technique instead of having a "Play" button available to view animations with a fixed speed. Playback with a fixed speed is sometimes slower than necessary, causing the user to waste time, and at other times can be too fast, requiring the user to rewind the animation. On the other hand, forcing the user to scrub manually means the user will always be navigating through time as fast as they can (or want), slowing down when they want or need to.

[1]http://ref.rufiange.com/diffani2013

The "unified dragging" technique that we implemented (shown in Figure 4) uses the same sideways drag to pan and to navigate in time. This simplifies the input required from the user, obviating the need to switch between mouse buttons and/or between slider widgets. No matter what kind of tiles are in the sequence, navigation through time is always done the same way, and does not require first pointing at a small target with the mouse cursor (an action that could require significant time [26]).

All of the combinations depicted in our taxonomy (Figure 3) are supported, and the mix of representations may be interactively changed by the user. As shown in Figures 1 and 5, the user may fluidly draw a stroke with their mouse to select one or several time slices (similar to drawing part of a lasso gesture), and then intersect their own ink trail (creating a pig-tail, inspired by Scriboli [22]), popping up a menu allowing the selected tiles to be converted to a different representation. This quick gesture allows the user to create any mix of the three visual representations (i.e., small multiples, diff or animation). Whenever the user interactively changes the representation of time slices, an animated transition illustrates the conversion of tiles (this animated transition is not to be confused with the animation of the graph shown within an animation tile).

The design choices we made for the real experiment (e.g., the "uni-

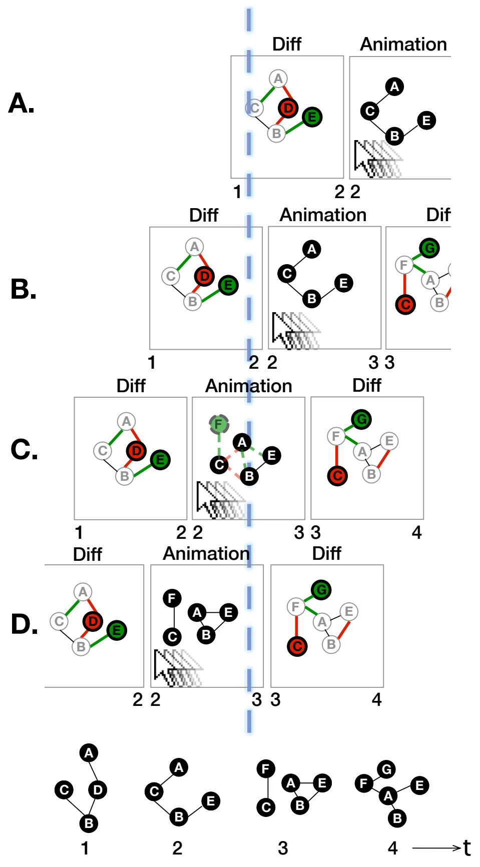

Fig. 4. Illustration of the unified dragging mechanism implemented in our prototype. The blue dashed line indicates the current time (t) of a dynamic graph (also shown using four small multiples at the bottom). When a user drags the mouse inside small multiple and diff tiles (e.g., **A**, **B**), the sequence of tiles moves in the same direction as the dragging. Animation tiles (e.g., **C**, **D**) do not move in space to allow the user to travel in time instead. **A**: $t \approx 1$. All the elements inside the tiles are at their initial positions. **B**: $t \approx 2$. The animation tile remains unchanged, but the sequence of tiles has moved to the left. **C**: $t \approx 2.5$. The time cursor is between two time slices inside the animation tile, and thus the positions and the transparency of the nodes are interpolated. **D**: $t \approx 3$. At the end of an animation tile, the nodes are at their final locations.

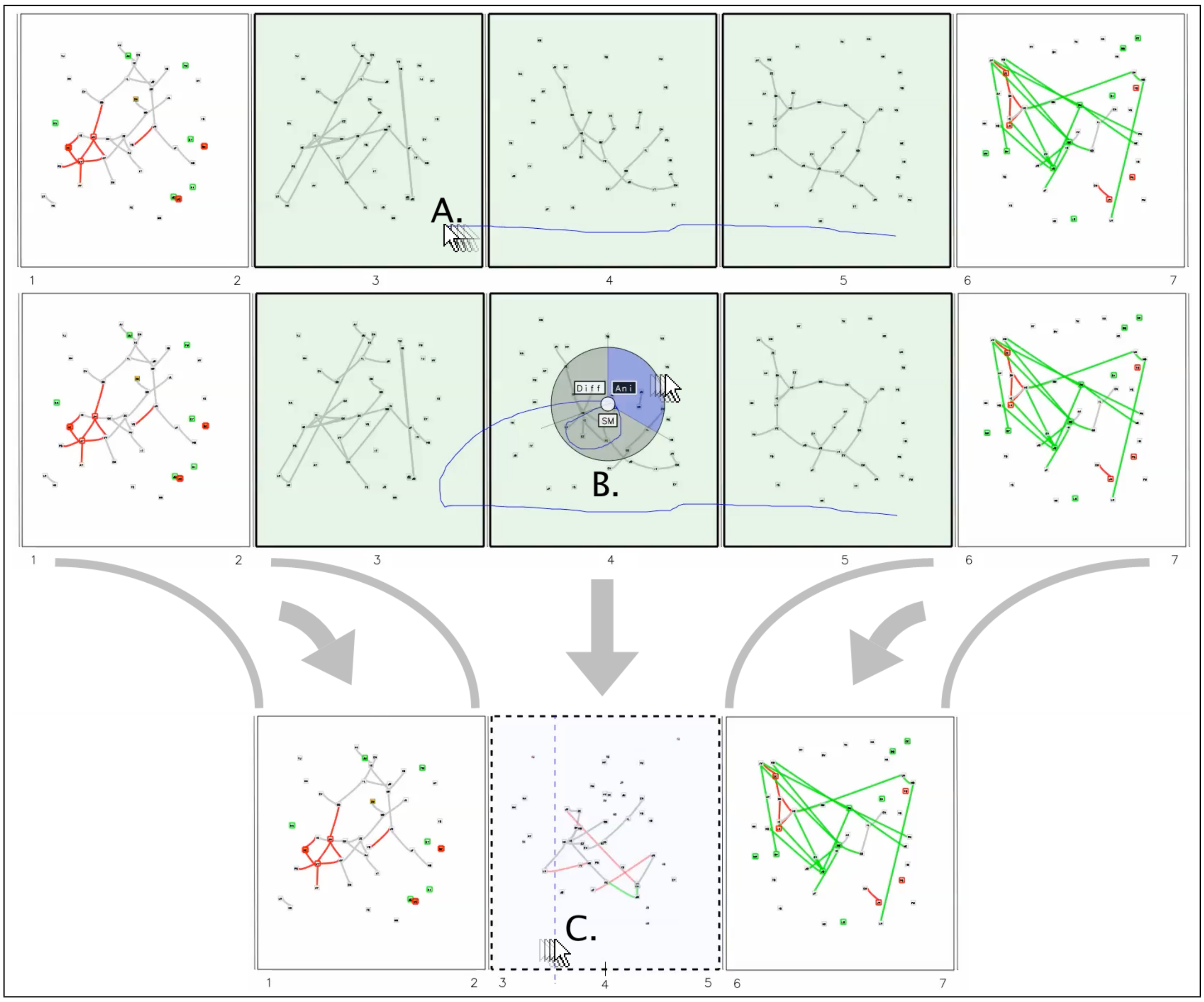

Fig. 5. Interactively converting three small multiple tiles into a single animation tile. Initially (top of figure), we have a diff tile followed by three small multiple tiles followed by a diff tile. ***A:*** Dragging the mouse to select a contiguous set of tiles covering time slices 3-5. ***B:*** When the user intersects their own ink trail, a "pig-tail" is detected and a radial menu pops up, allowing the user to change the representation of the selected tiles. ***C:*** The previously selected small multiples are compressed into a single animation tile (this conversion is shown with a smoothly animated transition).

fied dragging" technique) were based in part by the feedback we received in our pilot study.

5 Task-oriented Study

There are at least three theoretical sources of differences in performance that we can expect for the diff and animation techniques. First, with the diff technique, the user can compare two consecutive tiles with rapid eye movements, and even understand these two consecutive time slices by looking at the color highlighting within a *single* tile, whereas doing the same with the animation technique requires dragging with the mouse. Second, displaying N consecutive time slices with the animation technique is done with a single tile that is always visible, whereas the diff technique results in $N-1$ tiles that do not, generally, fit within the window, therefore requiring the user to perform some dragging to scroll through the tiles. Third, if there is significant movement of nodes from one time slice to the next, the diff technique could make it difficult to identify corresponding node positions in consecutive tiles, whereas the animation might make this clear by virtue of smoothly interpolating node positions. These three differences favor diff in the first case, and animation in the other two cases.

We cannot predict with certainty the net effect of these differences. However, we propose three hypotheses. First, we suspect that diff's advantage from fast eye movements will be strongest when there is little or no movement of nodes. Second, the relative performance of animation with respect to diff should improve when there is more movement of nodes. This second hypothesis is more speculative than the first, since performance with animation could also plausibly degrade, due to the user having to drag more *slowly* with faster moving nodes to monitor their individual changes. Finally, given the theoretical tradeoffs between diff and animation, we suspect that the hybrid mixture of them in DiffAni can sometimes result in better performance than with either of the non-hybrid techniques.

Our hypotheses are thus as follows :

- **H1**: with greater movement of nodes across time slices, the performance of diff degrades.
- **H2**: with greater movement of nodes across time slices, the performance of animation with respect to diff improves.
- **H3**: the DiffAni hybrid visualization lead to better performance than with non-hybrid ("pure") diff or animation.

To test these hypotheses, we generated three datasets. All three datasets contain a mix of transitions involving either no movement or much movement, as illustrated in Figure 6. In the first dataset (A: low movement), the nodes do not move between time slices 1 and 3, then do move in the next two transitions, then cease moving. The second dataset (B: high movement) has a longer lasting stage of movement in the middle. The third dataset (C: phased) alternates between movement and stabilization. The datasets were generated by adding and removing random numbers of nodes and edges at each time slice using the parameters in Table 1.

Table 1. Parameters used in the generation of the three datasets. Columns indicate average initial number of nodes and edges, average number of nodes added and removed in each transition, and average number of edges added and removed in each transition, respectively.

	$N(t=0)$	$E(t=0)$	$\Delta N+$	$\Delta N-$	$\Delta E+$	$\Delta E-$
Low	40.1	34.0	3.9	1.7	5.3	6.0
High	38.9	31.1	4.1	1.9	4.4	6.1
Phased	37.9	33.8	3.6	1.8	4.3	5.9

In normal usage, DiffAni allows users to select intervals of time and change the representation of a part of the dynamic network to small multiples, diff or animation. However, in our study, users were forced to visualize the network in one of three conditions : a "diff" condition which used only diff tiles, an "animation" condition which displayed the entire network in a single animation tile, and a "hybrid" condition which displayed low movement transitions with a diff tile and high movement transitions with an animation tile. Users could not interactively change the representation of any tiles in the experiment.

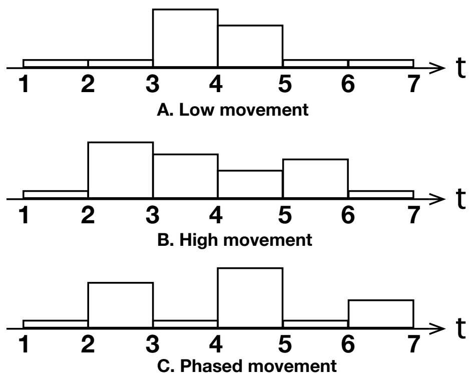

Fig. 6. Schematic representation of the degree of movement of nodes in the three datasets used in our evaluation.

The nodes in the datasets were positioned using a force-directed layout [13], computed for each time step of the dynamic graph. Differences in topology between the time slices naturally led to movements of nodes across time slices. Also, node positions were kept fixed for certain time steps, depending on the dataset (i.e., the transitions with a low degree of movement in Figure 6).

As shown in our taxonomy (Figure 3), a single diff tile can display the differences between two consecutive time slices or between two non-consecutive time slices. In the latter case, this serves to summarize several time slices in a single tile. However, in our experiment, the tasks required the user to examine every time slice and not just compare the first and last time slices. Therefore, the diff tiles always showed consecutive time slices. So, in the "diff" condition, there was a total of six diff tiles to show the seven time slices. In the "animation" condition, there was always a single animation tile, and in the "hybrid" condition, the number of tiles depended on the dataset. In particular, the hybrid representation had five tiles with low movement (four diffs and one animation), three tiles with high movement (two diffs and one animation), and six tiles with phased movement (three diffs and three animations).

Contrary to small multiples, diff can visualize changes between time steps, by combining two static representations and highlighting the differences. To place the nodes with the diff technique, we computed the averages of the starting and ending node positions for each transition of the evolving network. Another possible approach is to display several copies of the same nodes at once (e.g., [42]), but it can also make it more difficult to track individual nodes and consume more screen space.

We took inspiration from the set of questions used by Archambault et al. [1], and developed the following set of questions :

- **q1**: find which node is removed then reinserted most often over time.
- **q2**: find the node with the highest average degree over all time slices.
- **q3**: find the node whose degree never decreases.
- **q4**: find the path (chain of nodes) that is never disconnected in any time slice.

Comparing these tasks to Lee at al.'s taxonomy [25], q1 is concerned with the appearance/disappearance of nodes over time, and cannot be classified in Lee at al's taxonomy which was not designed with dynamic graphs in mind. However, we find that q2 and q3 are Topology/Adjacency tasks, and q4 is a Topology/Connectivity task. Our set of questions is also comparable to that used by [1], which included one Adjacency task, one Connectivity task, and two tasks focusing on the appearance/disappearance of graph elements. All questions were multiple-choice with four candidate answers. To make these questions more difficult, additional nodes and edges were randomly inserted and removed over time. In addition, participants could not simply look at the last time slice to determine the answers. For instance, in task q4, paths were randomly disconnected then reconnected later on.

5.1 Experimental design

In the user study, to verify our research hypotheses (H1,H2,H3), we asked participants to perform four tasks using three different interfaces. Our motivation was to explore possible compromises in using the techniques, depending notably on node movements. In particular, we wanted to check whether our hybrid approach could benefit from using both animation (e.g., to track moving nodes) and diff techniques (e.g., the scrolling in time is unnecessary if nodes are stable).

The twelve participants (2 female and 10 male) were students in computer science, and the presented hybrid approach was new to them. We generated datasets to test all the conditions and tasks, and also made sure there was only one possible answer for each question and that it wasn't too difficult but still realistic, by adding a random level of noise. In the experiment, users first performed warmup trials for each technique (excluded from our final results) where they answered two questions about a 4th dataset that contained phased movements.

At the start of each trial, an instance of one of the questions was displayed in the main window, allowing the user to take the time to read and understand the question before pressing a "start" button. Next, the network data was displayed. In the first time slice of the network, the four candidate nodes (or pairs) were highlighted. However, these candidates were not highlighted in any other time slice, contrary to [1].

The user could navigate by dragging until they determined the answer to the question, and entered the answer using radio buttons. If the first attempt by the user was wrong, they could retry up to two additional times, for a maximum of three attempts for each trial. Trials with three unsuccessful attempts were counted as errors. Users were instructed to complete trials as quickly as possible, without errors.

In the final experiment, we used a within-subjects design with a total of

3 techniques (diff, animation, hybrid)
$\times$ 3 datasets (low movement, high movement, phased movement)
$\times$ 4 questions
$\times$ 12 users
= 432 trials in total.

The order of presentation of techniques was counterbalanced with a Latin-square design, and the ordering of datasets and questions was random for each participant. We controlled node movements in the datasets, based on our assumption that movement can play a role in deciding whether diff or animation should be used to visualize parts of dynamic networks. To test the effect of node movements in the experiment, node positions were sometimes kept fixed for specific time slices, causing non-optimal layouts (as illustrated in Figure 5 in time slices 3 and 6-7).

We ensured that the difficulty and complexity of the tasks were reasonable by doing a pilot study, that was performed prior to the final experiment. The pilot study included eight participants, and motivated certain changes in the prototype. For instance, in a previous version, the user used a traditional scrollbar widget to navigate in time within animation tiles (as in [1]). Also, animations between consecutive time slices were displayed in three stages, to first show disappearing nodes and edges, then movements of nodes, then appearances of nodes and edges, following the staged animation approach of [27]. Both of these design choices seemed to penalize performance with animation. Furthermore, users seemed to be confused with using a scrollbar to navigate in animation tiles, while being able to drag anywhere to scroll through diff tiles. In the staged animations, some participants had difficulties distinguishing between the different slicings that were used in the timeline, i.e., slices to separate time steps, and also slices for each stage of an animation (placed between two time steps). Therefore, for the full study, we adopted the unified approach in Figure 4, where dragging anywhere serves to navigate both kinds of tiles. Our pilot study also tested a condition where all time slices were shown with small multiple tiles (without coloring of differences). However, since these small multiples were clearly inferior to the other techniques, they were excluded from the final experiment.

6 Results

In this section, we present the results of our experiment. The total time and error rates are shown in Tables 2 and 3. Shapiro-Francia tests on total times revealed them to not be normally distributed, hence Friedman tests were used to check for significant differences between visualization techniques for the different tasks and movement conditions. Over all movement types, the hybrid was found to be (weakly) statistically better than diff ($p < 0.10$). Examining only the average times, hybrid has the best total times in two of the movement conditions (indicated in bold in Table 2).

Table 2. Average duration of task trials for each technique (in seconds), and grouped by movement type. These include the time spent on 2nd and 3rd attempts when the user's 1st attempt was wrong.

	Low	High	Phased
Diff	135	140	128
Animation	117	147	**106**
Hybrid	**104**	**123**	128

We also checked whether some visualization techniques were harder (i.e., required more attempts) depending on the movement condition. Table 3 shows that the error rates (after 3 attempts) were below 5% in almost all cases, indicating that users were able to complete the tasks.

Table 3. Average number of attempts per trial (for all tasks). In parentheses are error rates, where a trial is considered an error if the user's 3rd and last attempt is still wrong.

	Low	High	Phased
Diff	1.13 (0.0%)	1.09 (6.4%)	1.00 (0.0%)
Animation	1.11 (2.1%)	1.15 (0.0%)	1.09 (2.1%)
Hybrid	1.09 (2.1%)	1.11 (2.1%)	1.15 (0.0%)

The participants made fewer attempts on average with the hybrid in the low movement condition, while the number of attempts was lower with diff in the other cases. However, with high movement, diff yielded a higher error rate than the other techniques. As for phased movement, diff resulted in no mistakes. Also, no significant differences were found ($p > 0.10$). These results indicate that the techniques were not very different from each other, in terms of number of attempts. Moreover, these results suggest that the hybrid approach was not more difficult to use than the non-hybrid techniques, although it required skills in both visualizations. Table 4 shows the total times broken down by task.

Table 4. Results for all interfaces, tasks and movements (times are in seconds). Within each task and movement combination, the minimum time is shown in bold (or two numbers are in bold, if they are not significantly different from each other). Two stars indicate the bold number(s) is (are) significantly ($p < 5\%$) smaller than the non-bold number(s), whereas one star indicates weak significance ($p < 10\%$).

Low movement	Task 1	Task 2	Task 3	Task 4
Diff	132	251	**45***	109
Animation	**107**	**219***	**54**	86
Hybrid	120	**160***	55	**80**
High movement	Task 1	Task 2	Task 3	Task 4
Diff	**123**	253	**63**	**120***
Animation	162	232	68	126
Hybrid	**134***	**206**	**49***	**100***
Phased movement	Task 1	Task 2	Task 3	Task 4
Diff	100	256	**34***	121
Animation	**96***	**198***	53	**76**
Hybrid	143	**207****	58	104

Regarding hypothesis H1, examining the averages, diff performed worse in the high movement condition than the low movement condition in three of the four tasks, indicating that collecting more data might confirm H1.

Concerning H2, when we compare low movement and high movement, we find the change in time for animation is always worse than the change in time for diff (i.e., the performance of animation decreases more than diff with higher movement). This contradicts H2. This may be because users had to drag very slowly in the high movement condition to be able to track node movements. Figure 7 shows the fraction of time spent dragging in each condition. As can be seen, the percentage of time spent dragging is higher with animation, compared to diff, in every condition.

H3 was partially confirmed : in the low movement condition, the hybrid was (weakly) significantly faster than diff for task 2 ($p < 0.10$), and it also had the lowest average for task 4. In the high movement condition, the hybrid was (weakly) significantly faster than animation for tasks 1, 3 and 4, and it was the best technique in terms of average time for three of the four tasks. With phased movement, DiffAni was significantly better than diff for task 2 ($p < 0.05$).

Comparing diff and animation in the low and high movement conditions, we find that diff seems less sensitive to movement, whereas performance with animation changes more between movement conditions. Interestingly, Purchase and Samra [28] found that user performance with animation varied non-linearly with the degree of preservation of mental map (which varies, roughly speaking, inversely with the degree of motion of nodes across time steps). These two results indicate that performance with animation varies in a complicated way, making it difficult for designers to choose a prescribed representation.

Analyzing the data by task, we notice that tasks 1 and 2 took more time than tasks 3 and 4. This is not surprising, since users could sometimes complete tasks 3 and 4 before examining all the candidate answers (e.g., in task 3, if a user noticed that a node's degree never decreased, he/she could try to immediately answer the question without examining the other candidates). Tasks 1 and 2, however, required that all the candidates be examined before answering. Furthermore, between tasks 1 and 2, task 2 required a more complicated mental cal-

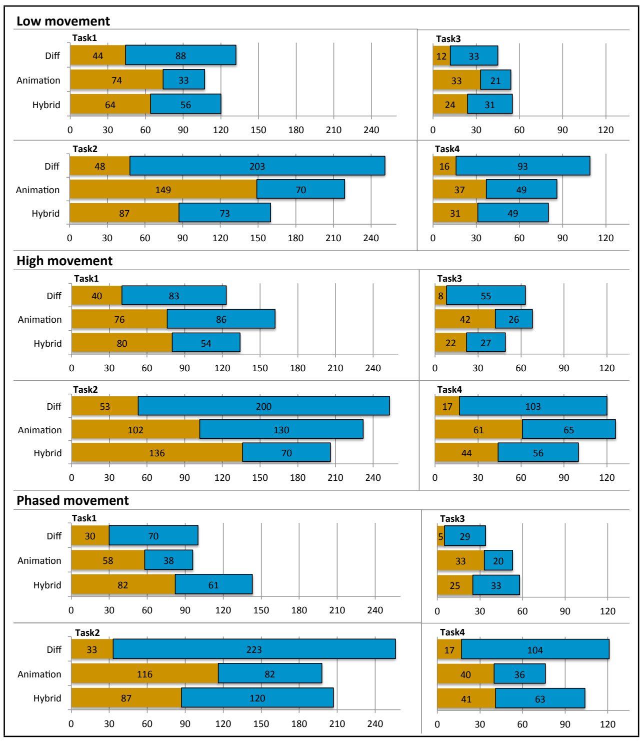

Fig. 7. Distribution of the time spent performing tasks using the visualization techniques, depending on the degree of movement (times are in seconds). The orange-colored bars indicate the participants were dragging in space (e.g., diff) or in time (e.g., animation), while the blue bars show the non-dragging time.

culation, and not surprisingly took the most time. Comparing tasks 3 and 4, task 3 was simpler and took less time than task 4 (which required examining entire paths of nodes).

In the first task (q1), animation was the fastest approach in the low movement condition, possibly because it facilitated the tracking of removed nodes (in Figure 7, the non-dragging time is lower). However, the performance of animation decreased with high movement. We suspect this is due to the fact that animation may not be the best technique to track new nodes, and even more so with increased movement, since they can appear anywhere. In this case, it may be faster to simply look for highlighted differences rather than scrolling in time with animation. Probably because of this more complicated trade-off, the hybrid approach performed average.

In the second task (q2), the hybrid visualization was the fastest technique with low and high movement. Looking at Figure 7, we can see it achieved significant gains in terms of non-dragging time (shown in blue). We suspect the unified dragging might have facilitated the tracking of nodes across representations. Since an animation is only used when there is movement, it is easier to track the nodes over time. Also, in such a hybrid representation, the starting and ending steps of a animated transition matches the previous and next node positions shown with a difference map (as shown in Figure 4).

In the third task (q3), participants were faster using diff, except in the high movement condition. Scrolling in time with animation may have been generally less effective for this task, compared with finding highlighted changes. We suspect that, since the degrees of some nodes were increasing over time, it might have made them a bit easier to locate. However, the tracking of nodes across time slices was probably more difficult with diff, especially with high movement. The hybrid approach, which used animation only sparingly and also integrated the unified dragging technique, performed better in this case.

In the last task (q4), the hybrid approach was generally faster, although it was slower than animation in the phased movement condition, possibly because of the mental effort required to switch representations very often. In this case, participants commented that animation generally helped them track nodes and chains of nodes across time slices. These results suggests that participants preferred animation to track nodes, but the usage of diff was also beneficial when nodes did not move much.

Examining average times for hybrid, within each task and across all movement conditions, we noticed that phased movement always yields the worst performance. For example, within task 1, hybrid took 143 seconds on average with phased movement, vs. 120 and 134 seconds in the other movement conditions. This indicates that the phased movement condition resulted in too many changes in the hybrid visualization; this is also corroborated by user feedback.

6.1 User feedback

In our user study, participants experimented with several approaches. In addition to collecting quantitative data (e.g., number of attempts, task durations, error rates), we also used five-point scales to evaluate their impressions.

The participants generally felt comfortable using the visualization techniques. In particular, animation and diff were rated as very easy to use (4.1 and 4.0, respectively), while the hybrid, perhaps because of the surprising interface and multiple representations, was evaluated at 3.6. However, it is interesting that participants also rated the hybrid as the most useful approach to perform tasks (4.3 vs. 3.9 for diff, and 4.0 for animation). We believe this correlates with other feedback we received that animation clearly helped track nodes efficiently, but only if used sparingly (as with the hybrid).

Several participants preferred our hybrid approach, arguing that tracking nodes was easier and that the use of animation was very useful and generally faster (P4,P5,P8,P9,P11-P13). Users further explained that they liked that the hybrid only uses an animation if a node actually moves over time (P4,P8,P11). However, some participants also said the performance of the hybrid decreased with alternating phases, mostly because they had to adapt to changing representations (P5,P8,P9). A few users felt that animation was not always the best technique, because they had to scroll in time too much (P7,P12), although another (P2) argued it was better to scroll in time in one big tile than across several tiles (sometimes causing a loss of context).

Participants also had suggestions for improvements, such as reducing cluttering (P1,P2,P6,P12) or highlighting nodes in several time slices (P4,P6,P9). They also believe that it could be useful to display tooltips showing, e.g., the number of neighbors of a node (P7,P9) or allow selecting nodes from a drop-down list (P4,P5,P9).

6.2 Applications

In this paper, we have shown that a hybrid technique can potentially help analyzing dynamic networks, but there are also practical uses of these combinations. In fact, to fully use hybrid approaches such as ours, one has first to construct a hybrid representation of the data and then analyze the resulting visualization. We focused on the latter case in our study. However, since users could benefit from mixing techniques (i.e., diff and animation) in some cases in our experiment, we propose a heuristic to construct hybrid representations of dynamic networks (illustrated in Figure 8).

First, a metric to evaluate the degree of node movements has to be computed for each transition (e.g., the average distance between time steps). Second, if there is no or little movement in a transition, diff can be used. Otherwise, animation was generally found to be more suitable. Also, if the same visual techniques are used for several consecutives transitions, they should be merged together, to avoid frequent switching in representations. Since small multiples performed significantly worse in our pilot study, it was not used in the real experiment. We therefore suggest that it should be only used for a small number of time slices, and also when the differences in transitions does not really matter.

In the spirit of previous work on hybrid visualizations (e.g., [21, 33]), a more suitable representation can be flexibly used for different parts of a dynamic network using our approach. For instance, in software engineering, refactorings can cause more instabilities (e.g., new

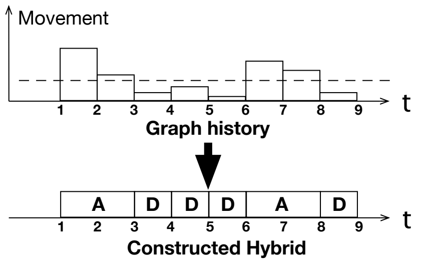

Fig. 8. A method to construct potential hybrids using our prototype. In this case, appropriate visualizations are chosen based on the degree of movement of nodes in the evolving network.

modules are added, moved or new connections are made) at the beginning of a project, and also once in a while in the software change history when the design is reworked [23]. However, bug fixes might not induce a lot of changes in terms of connectivities. Thus, the presented hybrid technique could help a software engineer understand how a design evolves over time, by using appropriate visualizations for different periods of time. As a software is usually developed over many years, this could allow engineers to browse large software histories faster.

There are also other possible applications of our hybrid approach that will be explored in future work. For instance, in a cell phone network, there are less movements at night, and depending on the time of the day, an appropriate visualization could be used. Also, in social networks, some connections do not change a lot while others can be more unstable (e.g., childhood friends vs. part time employees).

6.3 Limitations

We made choices in designing our experiment and prototype that we discuss in this subsection. We implemented a Fruchterman-Reingold algorithm [13] in our prototype, rather than possibly more complex strategies with unclear outcomes [28]. Also, more advanced layout algorithms are not always included in compatible software libraries or require expert knowledge. In our experiment, because of our choice of layout algorithm, we could not reproduce the exact same effects of mental map preservation which have been used in certain previous studies (e.g., [28, 29]), however we could indirectly vary mental map preservation, by varying the topology of the graph (i.e., adding and removing nodes and edges), which caused movement of nodes.

Our hybrid method relies on generic visualizations that can be used for any network. Also, the dynamic graphs in our datasets were randomized and had roughly between 35 and 70 nodes over seven time slices, similarly to previous studies (e.g., [1, 2]). However, user studies will be required to explore the possible effect of network density or the number of time slices on the performance of participants for more tasks and application domains.

In our experiment, the tasks chosen for the evaluation are based on existing taxonomies (e.g., [25]), but do not cover higher level changes (e.g., clustering, overview) nor attribute changes. We did not allow the highlighting of nodes over several time slices in our user study, to focus on comparing the visualization techniques themselves. Also, some tasks do not benefit from highlighting (e.g., finding a node, among many possibilities, that evolve according to some pattern).

We suspect a few uncontrollable factors could have contributed to reduce the statistical power of our results in addition to the small sample size. For instance, to use the hybrid fully, users needed to master two techniques, although some did not like or were less efficient with animation (while others liked it more than diff). To be fair, we let the participants practice the same amount of time using each technique, although users may have their own preferences and skills.

We used a fixed hybrid representation and did not allow the user to interactively construct hybrids in the experiment to limit the variability of our results (as done similarly in e.g., [1]). We focused on finding benefits on combining representations, and thus mixed typical and simple non-hybrid visualizations (e.g., diff, animation). However, the combination with other unexplored techniques (e.g., sparklines) could be beneficial. Also, the comparison of hybrid representations with other non-hybrids (such as multiple coordinated views [31]) should be explored in the future.

7 Conclusions and Future Directions

We have presented a hybrid visualization of dynamic graphs that allows the visual representation of the data to be varied across time. Multiple consecutive time slices in the data can be collapsed into a single tile displaying either an animation or a static difference map, enabling the user to choose trade-offs between space and time for different portions of the data.

We also proposed a taxonomy of visualizations of dynamic graphs (Figure 2) and a taxonomy of hybrid visualizations (Figure 3). Our prototype implementation allows users to flexibly explore these combinations.

Our experimental evaluation showed that the performance of different visualization techniques does not vary in a simple way, but that nevertheless the hybrid visualization can outperform other techniques in certain conditions, indicating that hybrids can result in a better trade-off than non-hybrid alternatives. We therefore recommend that hybrid visualizations allow users to interactively control the mixture of representations used for the data.

The user study differed from previous work in two ways that make it relevant for real-world expert use scenarios: first, the candidate nodes for tasks were not highlighted across all time slices, and second, navigation in time and space was performed with the same kind of mouse dragging (Figure 4) rather than requiring the user to use a small slider widget or play animations with a fixed speed.

One issue that could be investigated in future work is the reason why animation performed more poorly than expected, contradicting our hypothesis H2. We suspect that users may have been hindered by having to drag slowly during animations to be able to understand rapid node movements (and because of the added noise in the datasets). Future prototypes may improve performance by displaying "smeared out trails" or motion blur effects to make the movements of nodes clearer in fewer frames, and/or non-linearly vary the mouse cursor gain according to the speed of movements, to make it easier for the user to quickly understand transitions between time slices.

Another interesting research direction concerns the construction of hybrids. Although we presented a heuristic to design hybrids for dynamic networks, there is a need for more guidelines to determine how visualizations should be mixed in some context. Users that tried a few iterations of our prototype were generally faster with the hybrid, and thus it could be useful to study the learning curves of users to understand how to better exploit hybrid visualizations. Also, the interactive process of assembling hybrids itself could lead to discovering insights. Furthermore, alternative hybrid representations should be explored and evaluated to study, for instance, the effect of using different layout algorithms and interaction techniques.

Acknowledgments

Thanks to the participants in our study for their time, and to the other members of the HIFIV research group at ÉTS for their feedback. This research was supported by NSERC, and doctoral scholarships from Fonds de recherche du Québec and ÉTS. We also thank the anonymous reviewers for their valuable comments.

References

[1] D. Archambault, H. C. Purchase, and B. Pinaud. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE TVCG*, 17(4):539–552, 2011.

[2] D. Archambault, H. C. Purchase, and B. Pinaud. Difference map readability for dynamic graphs. In *Graph Drawing*, volume 6502 of *LNCS*, pages 50–61. Springer, 2011.

[3] J. Bertin. *Sémiologie graphique: Les diagrammes, Les réseaux, Les cartes*. Éditions Gauthier-Villars, Paris, 1967.

[4] U. Brandes and S. R. Corman. Visual unrolling of network evolution and the analysis of dynamic discourse. *Information Visualization*, 2(1):40–50, 2003.

[5] U. Brandes and B. Nick. Asymmetric relations in longitudinal social networks. *IEEE TVCG*, 17(12):2283–2290, 2011.

[6] M. Burch and S. Diehl. TimeRadarTrees: Visualizing dynamic compound digraphs. *Computer Graphics Forum*, 27(3):823–830, 2008.

[7] M. Burch, M. Fritz, F. Beck, and S. Diehl. TimeSpiderTrees: A novel visual metaphor for dynamic compound digraphs. In *Proc. Symposium on Visual Languages and Human-Centric Computing*, pages 168–175, 2010.

[8] M. Burch, C. Vehlow, F. Beck, S. Diehl, and D. Weiskopf. Parallel edge splatting for scalable dynamic graph visualization. *IEEE TVCG*, 17(12):2344–2353, 2011.

[9] F. Chevalier, P. Dragicevic, A. Bezerianos, and J.-D. Fekete. Using text animated transitions to support navigation in document histories. In *Proc. ACM CHI*, pages 683–692, 2010.

[10] J. A. Cottam, A. Lumsdaine, and C. Weaver. Watch this: A taxonomy for dynamic data visualization. In *Proc. IEEE VAST*, pages 193–202, 2012.

[11] M. Farrugia and A. Quigley. Effective temporal graph layout: A comparative study of animation versus static display methods. *Information Visualization*, 10(1):47 –64, 2011.

[12] P. Federico, W. Aigner, S. Miksch, F. Windhager, and L. Zenk. A visual analytics approach to dynamic social networks. In *Proc. I-KNOW*, pages 47:1–47:8. ACM, 2011.

[13] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[14] M. Gaertler and D. Wagner. A hybrid model for drawing dynamic and evolving graphs. In *Proc. GD*, pages 189–200, 2005.

[15] M. Ghoniem, J.-D. Fekete, and P. Castagliola. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information Visualization*, 4(2):114–135, 2005.

[16] M. Greilich, M. Burch, and S. Diehl. Visualizing the evolution of compound digraphs with TimeArcTrees. *Computer Graphics Forum*, 28(3):975–982, 2009.

[17] A. L. Griffin, A. M. MacEachren, F. Hardisty, E. Steiner, and B. Li. A comparison of animated maps with static small-multiple maps for visually identifying space-time clusters. *Annals of the Association of American Geographers*, 96(4):740–753, 2006.

[18] S. Hadlak, H.-J. Schulz, and H. Schumann. In situ exploration of large dynamic networks. *IEEE TVCG*, 17(12):2334–2343, 2011.

[19] J. Heer and G. G. Robertson. Animated transitions in statistical data graphics. *IEEE TVCG*, 13(6):1240–1247, 2007.

[20] N. Henry, A. Bezerianos, and J.-D. Fekete. Improving the readability of clustered social networks using node duplication. *IEEE TVCG*, 14(6):1317–1324, 2008.

[21] N. Henry, J.-D. Fekete, and M. J. McGuffin. NodeTrix: A hybrid visualization of social networks. *IEEE TVCG*, 13(6):1302–1309, 2007.

[22] K. Hinckley, P. Baudisch, G. Ramos, and F. Guimbretiere. Design and analysis of delimiters for selection-action pen gesture phrases in Scriboli. In *Proc. ACM CHI*, pages 451–460, 2005.

[23] I. Jacobson, G. Booch, and J. Rumbaugh. *The unified software development process*. Addison-Wesley, Reading, Mass, 1999.

[24] W. Javed and N. Elmqvist. Exploring the design space of composite visualization. In *Proc. PacificVis*, pages 1–8. IEEE, 2012.

[25] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proc. AVI BELIV Workshop*, pages 1–5. ACM, 2006.

[26] I. S. MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7(1):91–139, 1992.

[27] C. Plaisant, J. Grosjean, and B. B. Bederson. SpaceTree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *Proc. IEEE InfoVis*, pages 57–64, 2002.

[28] H. Purchase and A. Samra. Extremes are better: Investigating mental map preservation in dynamic graphs. In *Diagrammatic Representation and Inference*, volume 5223 of *LNCS*, pages 60–73. Springer, 2008.

[29] H. C. Purchase, E. Hoggan, and C. Görg. How important is the "mental map"? - an empirical investigation of a dynamic graph layout algorithm. In *Proc. GD*, pages 184–195, 2006.

[30] K. Reda, C. Tantipathananandh, A. Johnson, J. Leigh, and T. Berger-Wolf. Visualizing the evolution of community structures in dynamic social networks. *Computer Graphics Forum*, 30(3):1061–1070, 2011.

[31] J. C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Proc. Coordinated and Multiple Views in Exploratory Visualization*, pages 61–71. IEEE, 2007.

[32] M. Rosvall and C. T. Bergstrom. Mapping change in large networks. *PLoS ONE*, 5(1), 2010.

[33] S. Rufiange, M. J. McGuffin, and C. P. Fuhrman. TreeMatrix: A hybrid visualization of compound graphs. *Computer Graphics Forum*, 31(1):89–101, 2012.

[34] A. Sallaberry, C. Muelder, and K.-L. Ma. Clustering, visualizing, and navigating for large dynamic graphs. In *Graph Drawing*, volume 7704 of *LNCS*, pages 487–498. Springer, 2013.

[35] P. Saraiya, P. Lee, and C. North. Visualization of graphs with associated timeseries data. In *Proc. IEEE InfoVis*, pages 225–232, 2005.

[36] A. Telea and D. Auber. Code flows: Visualizing structural evolution of source code. *Computer Graphics Forum*, 27(3):831–838, 2008.

[37] B. Tversky, J. B. Morrison, and M. Betrancourt. Animation: can it facilitate? *International Journal of Human-Computer Studies*, 57(4):247–262, 2002.

[38] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. Fellner. Visual analysis of large graphs. In *EuroGraphics: State of the Art Report*, 2010.

[39] M. Wattenberg. Arc diagrams: Visualizing structure in strings. In *Proc. IEEE InfoVis*, pages 110–116, 2002.

[40] F. Windhager, L. Zenk, and P. Federico. Visual enterprise network analytics - visualizing organizational change. *Procedia - Social and Behavioral Sciences*, 22:59 – 68, 2011.

[41] J. S. Yi, N. Elmqvist, and S. Lee. TimeMatrix: Analyzing temporal social networks using interactive matrix-based visualizations. *International Journal of Human-Computer Interaction*, 26(11-12):1031–1051, 2010.

[42] 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. GI*, pages 183–190, 2011.

[43] S. Zhao, M. J. McGuffin, and M. H. Chignell. Elastic hierarchies: Combining treemaps and node-link diagrams. In *Proc. IEEE InfoVis*, pages 57–64, 2005.