Expressive Authoring of Node-Link Diagrams with Graphies

Hugo Romat, Caroline Appert, Emmanuel Pietriga

To cite this version:

Hugo Romat, Caroline Appert, Emmanuel Pietriga. Expressive Authoring of Node-Link Diagrams with Graphies. IEEE Transactions on Visualization and Computer Graphics, 2021, 27 (4), pp.2329-2340. 10.1109/TVCG.2019.2950932 . hal-02358999

HAL Id: hal-02358999
https://inria.hal.science/hal-02358999

Submitted on 12 Nov 2019

Expressive Authoring of Node-Link Diagrams with Graphies

Hugo Romat, Caroline Appert and Emmanuel Pietriga

Abstract— Expressive design environments enable visualization designers not only to specify chart types and visual mappings, but also to customize individual graphical marks, as they would in a vector graphics drawing tool. Prior work has mainly investigated how to support the expressive design of a wide range of charts generated from tabular data: bar charts, scatterplots, maps, *etc.* We focus here on an expressive design environment for node-link diagrams generated from multivariate networks. Such data structures raise specific challenges and opportunities in terms of visual design and interactive authoring. We discuss those specificities and describe the user-centered design process that led to Graphies, a prototype environment for expressive node-link diagram authoring. We then report on a study in which participants successfully reproduced several expressive designs, and created their own designs as well.

Index Terms—Expressive Design, Node-Link Diagram, Multivariate Networks

1 Introduction

The design of a visualization is primarily driven by decisions about chart type and visual mappings, which focus essentially on perceptual effectiveness. But when the primary purpose of a visualization is to communicate a message and engage its audience [39], designers will adopt a more creative process. This process will include considerations about, *e.g.*, memorability and aesthetics [26], which will uniquely influence the decisions made by each designer.

Expressive design approaches bring flexibility to this creative process, enabling designers to easily customize their visualizations and quickly test alternatives. One key issue to address is to remove the artificial boundary between the specification of the visualization and its interactive customization. Expressive design approaches achieve this by either integrating both activities in a single environment [26, 27, 39] or building bridges between them [10]. The resulting tools support a wide range of charts, as long as such charts can be derived from tabular data. They offer little or no support for multivariate networks [31], however. Designers who want to create expressive network visualizations must resort to dedicated visualization and analysis software and then customize the output in a general-purpose vector graphics editor. The former (*e.g.*, [3, 7, 32]) are good *generative tools* [10], but they fail to support the flexible, iterative design process that is key to *expressive* visualization authoring.

We introduce Graphies,[1] an expressive design environment for multivariate network visualization. Aimed primarily at authoring visualizations for communication purposes, Graphies focuses on node-link diagram representations of networks (Figure 1). Informed by a user-centered design process involving data analysts, Graphies brings together a coherent set of techniques essentially drawn from the literature, and integrates them in a flexible authoring workflow. The workflow itself is key to our approach, as it enables designers to freely intertwine all types of action: populating the representation, declaring visual mappings, customizing individual elements by direct manipulation, adding annotations, creating animations. Support for such intertwining streamlines the creative process [9, 27], effectively enabling quick design iterations.

After giving an overview of related work, we describe our user-centered design process and introduce a set of requirements derived from it. We then present Graphies' authoring workflow and interaction model, as well as its main features, discussing how they address those requirements. Next, we report on a first-use study in which participants successfully reproduced several expressive designs, and created their own designs as well. To the best of our knowledge, this study is the first to evaluate an *expressive* design environment for node-link diagrams. Based on the results of this study, we conclude with design guidelines and discuss possible avenues for future work.

- *Hugo Romat is with Univ. Paris-Sud, CNRS, INRIA, Université Paris-Saclay & TecKnowMetrix.*
- *Caroline Appert is with Univ. Paris-Sud, CNRS, INRIA, Université Paris-Saclay.*
- *Emmanuel Pietriga is with Univ. Paris-Sud, CNRS, INRIA, Université Paris-Saclay.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

[1] `https://hugoromat.github.io/graphies`

2 Related Work

The literature about network visualization is rich. We limit our overview to the most related systems for general-purpose expressive visualization design on one side, and for network visual analysis and visualization authoring on the other side, as we draw from both research areas.

2.1 General-purpose Expressive Visualization Design

In their taxonomy, Grammel *et al.* [16] identify the *template editor* and the *visual builder* as the best user interface schemes for creating visualizations. By using templates as starting points, editors such as Tableau [47] or ManyEyes [50] allow users to quickly design visuals. However, template-based approaches offer limited support for customization [39] in comparison to visual builders. With the latter, users have more control over how the graphical marks are assembled, and how data attributes are mapped to visual properties.

Mendez *et al.* [29] compare Tableau's classic *top-down* approach with iVolver's [30] *bottom-up* (or *constructive* [22]) approach. The top-down approach of Tableau is observed to be faster, but less transparent and less flexible, thus less adapted to the design of personalized, creative visuals. On the contrary, the bottom-up approach of iVolver is observed to encourage users to explore alternative visual designs.

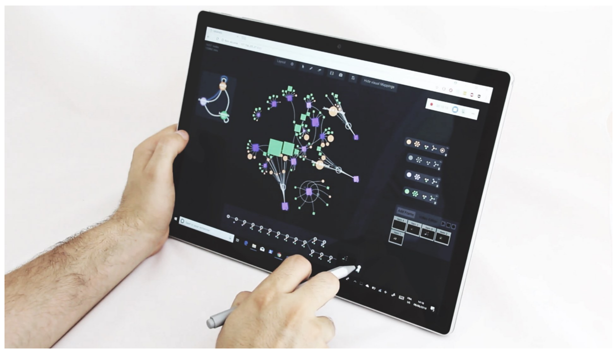

Fig. 1. Overview of Graphies' Web-based interface, running on a tablet with support for pen+touch.

Recent work has focused on environments that explicitly support *expressive* design. Prominent examples include Lyra [39] and Data Illustrator [27]. Users associate data with initially-agnostic vector glyphs, thus avoiding too much premature commitment [18] in the creative process. The two differ in their interaction model for mapping data to visual variables. Lyra works with drag-and-drop actions from data to interactive placeholders on glyphs, while Data Illustrator integrates data bindings as possible values for glyph (or layout) properties, that users can select in inspectors. Charticulator [35] follows the same general approach but considers layout as a driving element in the authoring process. With Data-driven guides [26], designers create freeform illustrations in a vector graphics editing environment, and can bind geometric properties of those shapes (length, position and area) to data attributes.

These *expressive authoring tools* offer much more flexibility than *generative tools* (*e.g.*, programming frameworks such as D3 [11], or business intelligence products such as Tableau [47]), the latter often requiring designers to resort to vector graphics drawing tools to manually customize their output [10]. However, while expressive tools enable the creation of a wide range of visualizations, they work with tabular data, and have little or no support for network data structures. Those that do, such as iVisDesigner [34] or Charticulator [35], are forced to expose the concepts of nodes and links in an indirect, somewhat cumbersome manner, failing to explicitly represent the multivariate data in the context of the network's topology [31].

2.2 Authoring Graph Visualizations

While the above environments make it possible to create a wide range of visualizations, other systems are specifically designed for working with networks and coping with the specific challenges they raise. Indeed, representing the topological structure and the attributes associated with the nodes and links of multivariate graphs is often difficult [31]. For instance, spatial position is typically under the control of the graph layout algorithm, which tries to produce a representation that is legible, based on heuristics such as minimizing link crossings [52]. As a consequence, node size variations are fairly constrained, eventually limiting the number of channels for the visual mapping of data attributes [37,38]. The problem gets more difficult as graphs grow in size and complexity [31,51] and when they are dynamic [25].

Some systems operate at a very low-level: GraphViz [15] consists of command-line programs taking graph data files as input, that users can edit in any text editor to change the appearance of nodes and links. GUESS [2] provides users with an interactive interpreter in which they can perform selection and filtering operations on the loaded data, and specify visual mappings by means of code instructions. D3 [11] provides developers with a full-featured API to generate interactive graph visualizations on the Web. Still on the Web, GraphCoiffure [46] is a system that lets users apply style sheets to static graphs. These systems have much expressive power but they all require programming skills. Users cannot edit the visual representation directly, which severely limits the accessibility of those tools [26, 39] and does not favor an iterative design process.

NodeXL [44,45], as a Microsoft Excel plug-in for the creation of node-link diagrams, is accessible to a broader audience. NodeXL lets users specify mappings from data attributes to several visual variables. Users benefit from their experience with the well-known spreadsheet program, but NodeXL has to comply with Excel's model for producing charts: users have to work conjointly with raw data in a tabular form and with the graphics. Again, direct editing of the visual representation is limited.

Designed for data-driven storytelling, DataToon [25] has some commonalities with Graphies (further discussed in Section 4.1). DataToon has a different focus, however: helping design comics-inspired, paginated representations of relatively small dynamic networks [4]. As such, it follows a top-down approach: users populate the workspace by choosing node & link types of interest, adding all instances at once and filtering them *a posteriori* using direct manipulation. While Graphies supports this approach, it also supports a bottom-up one: users can first filter out the data based on type and attribute values, before actually putting them in the workspace. The two strategies coexist, letting users incrementally add elements with one or the other. This bi-directionality enables Graphies to offer the simplicity of direct manipulation while scaling to larger networks.

2.3 Visual Exploration of Multivariate Graphs

Beyond the above systems, that are focused on authoring network visualizations, there is also a variety of tools for their visual analysis. We focus here on systems that provide effective support for the visualization of multivariate networks, and refer the reader to Nobre *et al.*'s recent survey [31] for a complete picture of the state of the art. One possibility consists in forcing specific layouts by mapping up to two node attributes to spatial position. Both Semantic substrates [43] and PivotGraph [53] lay out nodes in a scatterplot-like manner. This makes it possible to encode attributes with position, but can also adversely impact the graphs' legibility as node placement is no longer optimized with respect to the earlier-mentioned legibility heuristics. This calls for some interaction in order to enable users to switch between different projected views, an idea that was further explored in GraphDice [8].

The Network Lens [23], as all systems that adopt a focus+context strategy, heavily relies on interaction. Users move a lens on top of the graph represented as a basic node-link diagram to reveal plots associated with its nodes' attributes. Another option consists of aggregating nodes using different clustering strategies based not only on topology, but on attribute values as well [1,33,41].

GraphTrail [14] takes a radically different approach, focusing on the multi-variate data associated with the network's elements, that get displayed using charts that bin nodes and links according to selected attributes. Users can get an overview of the network by juxtaposing several such charts. The idea of hiding the topology to the benefit of attribute-based statistical charts had already been explored in NetLens [24], but was limited to a specific meta-model at the time. DOSA [48] explores a compromise between this approach and node-link diagrams, embedding the charts into the node-link diagram to represent aggregated nodes.

Systems such as Gephi [7], Tulip [3] and Pajek [32] are designed to support the exploration of large networks. They typically have a steep learning curve, as they provide users with a rich set of features: filtering, aggregation and analysis functions, as well as presentation functions. Users can change the appearance of network elements by specifying basic visual mappings. However, these apply globally, limiting flexibility as elements cannot be edited individually by direct manipulation. The workflow is primarily intended to support data analysis, and the associated interaction model is often indirect and complex [8,21]. Again, the flexibility and accessibility of the design workflow is limited.

In summary: on one hand, general-purpose expressive visualization environments are accessible solutions that bring much flexibility to the design workflow, but that are not well-adapted to multivariate networks; and on the other hand, network visualization software are well adapted to these particular data structures, but feature very limited support for expressive design. Our goal with Graphies is to reconcile both, so as to ease the expressive design of node-link diagrams from multivariate networks.

3 Design process

We have been working on the design of Graphies over the course of one year with data analysts from a consulting company. The company provides services such as making cartographies of innovation in various domains of activity, by surveying and monitoring the scientific and technological production of the different actors in these domains. The analysts' work is supported by a large database that contains multiple types of resources (actors, scientific articles, patents, technical documents) and multiple types of relations (authorship, collaboration, citation). Based on these data, which form a large, multivariate, and heterogeneous network [31], analysts can picture a complex domain, and advise their clients about opportunities for innovation. Depending on their clients' needs, they work on tasks such as, *e.g.*, identifying the main actors in a given domain, how different domains are articulated, what domains are active in academia and industry. Essentially, they have to find insights in the data, and present those findings to their clients, which often involves creating node-link diagrams.

The first author working in close collaboration with the above-mentioned company, we had the opportunity to involve data analysts regularly in the iterative development of Graphies. We conducted a longitudinal observation that helped us understand their tasks and workflow, the tools they use, how they use them, and what sort of problems they face. These observations informed the initial design of Graphies. We then conducted five sessions of demonstrations and interviews regularly distributed throughout the design process. During those sessions, data analysts played with the latest iteration on the Graphies prototype. We gathered feedback about 1) the features that we had already implemented, 2) the features that were under development, and 3) the features that data analysts wanted us to include.

Our first high-level observation is consistent with what Spritzer *et al.* report about journalists [46]. The typical network visualization construction workflow of these expert users involves at least two types of tools: 1) a graph analysis tool such as Pajek or Gephi to delimit the subgraph of interest and apply coarse-grained visual mappings to it (such as setting node size based on a data attribute, color on another, and showing node labels); and 2) a vector drawing editor such as Adobe Illustrator or even Microsoft Powerpoint to further edit the diagram: making manual adjustments to the layout and visual variables of some nodes and links to emphasize them, adding annotations, improving the overall visual design. This also echoes the observations reported by Bigelow *et al.* in their study of *"how designers design with data"* [9], as well as the general observations made about design workflows in recent work about expressive visualization environments [10, 25–27, 39]. This kind of two-step workflow that involves two independent tools suffers from multiple usability problems, as mentioned already.

The main problem is the artificial, and sometimes fuzzy, boundary imposed by the use of two standalone applications. When moving from the graph analysis tool to the vector-graphics editor, relations between visual marks and data are lost [10]. Any small modification that involves updating visual variables based on data attribute values entails going back to the first tool, regenerating the raw diagram, and then redoing all personalization edits in the second tool [9]. Another problem in the case of node-link diagrams is that, topological information having been lost in the vector graphics editor, editing operations in the personalization phase that rely on this information, such as, *e.g.*, adjusting the topology, or bundling and fanning edges, require tedious low-level geometry editing. In addition, a striking observation we made was how often users have to go back and forth between the visualization and the raw data in the graph analysis tool. This is consistent with Grammel *et al.*'s study [17], who observed that users switch back and forth between visual mappings and data attribute selection when constructing a visualization.

Relating our observations to Green's cognitive dimensions framework [18], we see that our users' design process is impeded mostly by significant *premature commitment* and by the *viscosity* associated with any small change, which is particularly pronounced when handling node-link diagrams. Based on these observations, we identified the following set of initial design requirements:

- R_1: Avoid artificial boundaries between the data encoding stage and the visual design stage [27]: populating the canvas with nodes and links; specifying visual mappings; manually adjusting the layout and appearance of individual elements; adding static content such as annotations. All such actions should be seamlessly integrated into the same workspace, enabling designers to interleave them at will.

- R_2: Enable designers to perform those operations using rapid, incremental, reversible actions whose effects on the objects of interest are visible immediately, following the principles of direct manipulation [42].

- R_3: Enable designers to specify a wide range of on-node and on-edge encodings [31], for both categorical and numerical attributes.

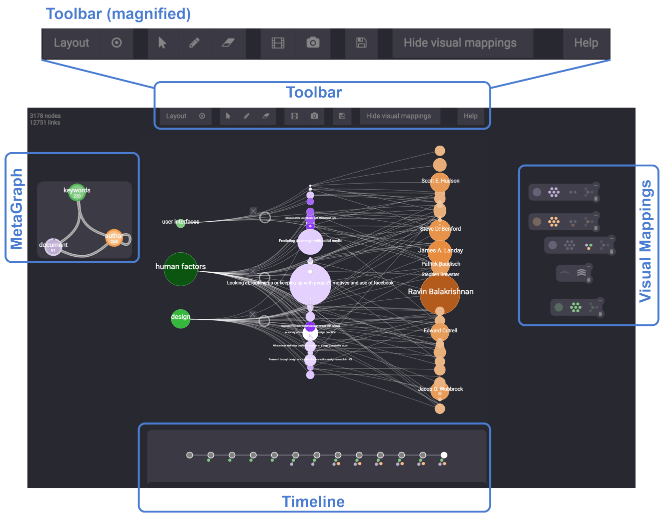

Fig. 2. Graphies: main user interface components. The MetaGraph, Visual Mappings and Timeline are organized on independent layers.

In one of the later sessions with data analysts, our prototype had reached a sufficient level of maturity to conduct a preliminary evaluation. We asked three data analysts to perform a series of three specific tasks, first with their usual set of tools, then with Graphies. We worked in collaboration with a fourth analyst so as to make the tasks representative of their work. The other three data analysts were then asked to explore a dataset to answer a question that a client could have, and subsequently design a visualization to present their findings. The task was over when participants were satisfied with the node-link diagram they had made. We encouraged them to express out loud their actions when using Graphies. We also conducted a semi-structured interview at the end of the session to capture their general impressions. This preliminary study yielded encouraging results: all participants were satisfied with the produced visualizations, and never felt the need to resort to external tools. They particularly appreciated the ability to access data *in-situ*, directly from the canvas where they were designing their visualization (R_1), and how rich (R_3) and easy-to-explore (R_2) the design space of visual encodings was.

At different stages during these sessions, data analysts expressed the need for additional features, which we took into consideration at various stages in the development of Graphies. For instance, in situations where they had to deal with networks similar in structure and content to other networks they had created visualizations of before, they wanted to be able to reuse those existing visualizations as starting points. We further discuss this feature in Section 4.4. They also made a set of related feature requests: be able to easily explore different designs and compare them; be able to animate between the stages of a design to better understand the differences between them; make it possible to export series of diagrams for the same network, arranged into an image gallery for storytelling purposes [25]. We derived the following two additional requirements from these feature requests:

- R_4: Going beyond support for basic undo (R_2), enable multiple visualizations of the same network to co-exist, and let users easily fork and switch between them to explore alternative designs.

- R_5: Enable smooth, animated transitions between these different visualizations, both as a means to help users keep track of changes and to produce animated node-link diagrams [5] that can support a rudimentary form of network storytelling [12].

4 Graphies

As an expressive design tool, Graphies is primarily aimed at supporting the authoring of node-link diagrams made for communication purposes. In such contexts, users often know what elements or types of elements they want to show. They also have some idea about how they want to show them, but need to be able to quickly test different options. The core challenge is not to display all nodes and links, but rather to let designers build personalized views on subsets of the data using a flexible, iterative workflow.

Figure 2 shows Graphies' interface. Its main component is a zoomable canvas that holds the diagram. The MetaGraph (left) provides a summary of the full dataset (showing the different types of nodes and how they are connected), allowing users to incrementally populate the main canvas by drag-and-drop. All interactions to modify the nodes' and links' visual appearance and spatial layout (R_2) are performed by direct manipulation in the canvas or by invoking contextual widgets (see Figures 4 & 6). The Toolbar features icons that trigger global actions: optimize the layout, toggle annotation mode, export the diagram as pictures or video (R_5). Finally, a Timeline enables users to revert back to any past state of the diagram, on any branch (R_4).

Implemented as a Web application, Graphies runs on a variety of devices: workstations equipped with mouse and keyboard, multi-touch devices, devices with support for pen + touch. On tactile surfaces, all navigation actions and manipulations of graphical elements can be triggered with touch gestures. When the device supports pen input, all interactions (including text entry) can be performed directly on screen (Figure 1). Such pen + touch environments are particularly well suited to designing node-link diagrams, as they make precise, arbitrarily-shaped selections of network elements easy to perform. They also provide good support for free-form annotations, which can be particularly useful when designing for communication purposes.

4.1 Populating the Canvas with Data

Users can import networks from JSON files, or load previously-saved diagrams. The MetaGraph gets automatically populated with the different types of nodes and links declared in the network: for instance, in Figure 2, which shows HCI publication data, nodes are of type: document, keyword and author. Each node (resp. link) in the MetaGraph is thus an aggregated representation of all nodes (resp. links) of the corresponding type in the dataset.

The MetaGraph allows users to populate the canvas without requiring them to interact with the raw data. It takes inspiration from OntoVis [40], while adding support for multivariate edges, and filtering. Similar to DataToon [25], the basic interaction consists of dragging a node or link type from the MetaGraph to the canvas to populate it with the corresponding elements. But unlike DataToon, which only enables *a posteriori* filtering of elements by direct selection after they have been dropped on the canvas, Graphies lets users adopt a query-first strategy [31] by declaring restrictions on attribute values *before* dragging elements. For instance, clicking on meta-node keyword in Figure 3 enables users to restrict the selection to those that are most frequently used (here, $\geq$ 200 times). Once a filter is defined, users can drag either the whole result list or individual items to the canvas. Such *a priori* declarative filtering enables designers to deal with larger, more complex graphs featuring multiple attributes, both categorical and numerical (R_3).

Dragging and dropping meta-nodes in the canvas only populates it with the corresponding nodes. To add links, users then have to drag-and-drop meta-links. They can be dropped on a node selection, in which case the canvas will be populated with the subset of links that have their source or target in that selection. Dropping on one specific node has a similar effect, adding links to this node only. This lets users start from a node of interest and make the graph expand incrementally, in the spirit of the *"Search, Show Context, Expand on Demand"* navigation paradigm [49]. Filters can be set on meta-links in the same manner as on meta-nodes.

This interaction model brings flexibility, letting users adopt either a bottom-up approach to the construction of their diagram by populating the canvas only with elements of interest after filtering; or a more classic top-down approach by populating the canvas with all elements first, and then defining filters for elements to be removed, as in [25].

4.2 Customizing the Layout

Node and link placement is a central concern when designing node-link diagrams. Graphies relies on D3's implementation of the Barnes–Hut force-calculation algorithm [6] to optimize the graph layout. Force-directed layout algorithms have the good property of supporting incremental modifications to the graph. Users can drag and drop new nodes and links into the canvas, or remove existing ones, and the layout will smoothly adapt to these topological changes. The layout can be frozen by stopping an ongoing simulation, and the spatial density can be adjusted using a slider that controls the node repulsion force in the simulation. At any moment, users can also ask Graphies to optimize the layout again, with the option of forcing nodes into a radial layout (see, *e.g.*, Figure 5-c). Finally, they can adjust the position of any node freely by direct manipulation on the canvas (R_2). They can pin nodes as well, thus preventing the force simulation from moving them in subsequent runs.

Pinning is one of several commands exposed in the contextual toolbar that pops up when clicking on a node or making a lasso selection. The list of commands depends on whether the underlying selection consists of nodes (Figure 4-a) or links only (Figure 4-b). Beyond aggregate, pin and delete commands, the toolbar also makes it possible to further refine the selection. Indeed, some diagrams can make it extremely tedious to select only nodes or links of a given type in a particular region depending on the layout. With selection refinement, users can first delineate the region of interest on the canvas, regardless of the type of elements that lie inside, and then restrict the selection within those bounds to one type only.

Link-centric commands include interactive bundling [36] using a handle to manipulate the attenuation circle that parameterizes the bundle's attraction force and location (R_2). In Figure 2, outgoing links from the three most-frequently-used keyword nodes have been bundled. In the same spirit, Graphies also makes it possible to interactively fan links [36] connected to a node, *i.e.*, to radially distribute them in a uniform way to decrease clutter.

4.3 Creating Visual Mappings

Visualizations of multivariate networks based on a node-link layout typically show a subset of the data attributes with on-node and on-edge encodings [31] using visual channels such as, *e.g.*, color, size, shape, or stroke width.

Figure 6 illustrates how such mappings can be defined with Graphies. A long-press on a node or on a link pops up the visual mapping widget. The first step consists of specifying the scope of the mappings that will be defined next. For nodes, the scope can be set (Figure 6-a) to: the node on which the widget was invoked; all nodes of the same type; all nodes regardless of their type; the selected node's neighborhood; all nodes in active lasso selections on the canvas. Setting the scope then reveals the full visual mapping widget, as shown in Figure 6-b.

Data attributes are listed on the left, and visual variables on the right, along with domain and range information, respectively. Users declare visual mappings by connecting a data attribute with a visual variable. Both the domain and range of a mapping can be adjusted using interactive sliders and color selectors that behave according to direct manipulation principles. Changes made to visual mappings are directly propagated to the elements in the canvas, providing immediate visual feedback to users (R_2). Visual variables that are not linked to a data attribute have the same value for all elements that fall in the mapping's scope. This value can be adjusted using the same type of direct manipulation.

Graphies differentiates between categorical and numerical data attributes. When the attribute is numerical, the mapping between domain and range uses linear interpolation (in HSL space for color). When it is categorical, attribute values are mapped to discrete values evenly distributed in the visual variable's range (choosing from predefined schemes in the case of color hue).

Graphies supports a rich set of visual variables, giving users much flexibility in how they can represent the data (R_3). Users can map node attributes to the following visual variables: spatial position,

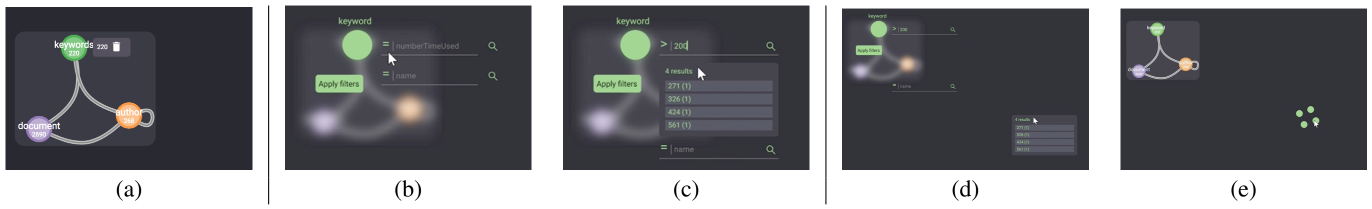

Fig. 3. Applying a filter to nodes of type *keyword*: (a) there are initially 220 such nodes; (b-c) only selecting nodes whose attribute $numberTimeUsed > 200$ yields 4 results; (d-e) dragging-and-dropping those 4 nodes on the canvas.

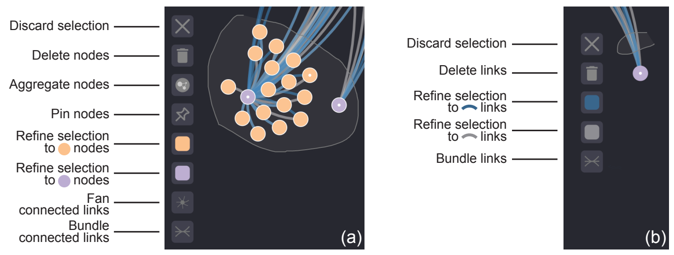

Fig. 4. Contextual toolbars for node and link selections.

size, stroke width, stroke color, fill color, shape, label size and color. Mapping attributes to spatial position makes it straightforward to create attribute-driven layouts [31, 43, 53]. For instance, nodes in Figure 2 are arranged in columns by mapping `type` (a categorical attribute) to the node's `x`-coordinate. Label size and color are managed as visual node properties. By default, labels are shown only when hovering. They can be made always visible by setting the label size to a constant value. Finally, node shape can be chosen among a set of basic geometries: circle, square, triangle. It can also be sketched, in the spirit of [25, 54] (Figures 5-b & 7), or replaced by a bitmap (Figure 5-a).

Similarly, users can map data attributes to link variables: stroke width, stroke color, curvature (which has recently been proposed as a means to encode data on links [36]), and texture. The latter can be based on a sketch or a bitmap. In each case, the texture consists of the input rendered in a repeating pattern along the link.

The variety of possible visual mappings on both nodes and links means that designers might want to create a legend to accompany their diagram. This is particularly relevant in the case of expressive designs intended for communication purposes. As creating such legends can be a tedious task, Graphies features a legend generator (see Figure 7) that automatically builds a summary of visual mappings that can later be manually edited. Mappings that have a global scope come first, followed by mappings that apply to a subset of elements only. The legend can also serve as a quick selection tool: clicking an item will select all nodes or links the corresponding visual mapping applies to.

4.4 Supporting an Iterative Design Process

A key characteristic of expressive visualization design environments is not only to provide designers with a wide range of options to visually encode data, but to enable a flexible design workflow as well. The environment should let designers quickly try multiple alternatives (R_4) and iterate on their diagram, in an *exploratory design process*.

As discussed earlier, this is primarily supported by enabling designers to perform all steps of the workflow in a single environment (R_1) without imposing artificial divisions between, *e.g.*, the data encoding stage and the visual design stage [27]. But additional features contribute to facilitating an iterative design process as well.

One important design choice we made in Graphies was about the *visual mapping evaluation strategy*. Visual mappings can be seen as rules that apply a set of styling instructions to a selection of elements as in, *e.g.*, the Cascading Style Sheet language (CSS). The two main strategies to handle such sets of rules in Graphies were: a) to always enforce mappings; or b) to apply them on-demand only. While strategy (a) will typically yield a higher level of consistency, it does this by restricting the designers' options, or resolving conflicts, or both. Strategy (b), on the other hand, applies mappings blindly, without imposing any restriction. However, it will also override values previously set by other visual mappings that apply to (some of) the same network elements. As flexibility is key to expressive design, we eventually opted for strategy (b). Active mappings are not enforced continuously by the system, but rather applied to the associated selection of nodes or links on-demand, implicitly, when designers interact with the corresponding widget. As it is important to be able to adjust the mappings, they can be accessed quickly from an independent layer drawn on top of the canvas. All mappings reside in this layer indefinitely (unless explicitly deleted), even if they conflict with other mappings. That layer can be hidden, or the mappings can be minimized individually to limit clutter.

Beyond the adjustment of existing mappings by direct manipulation, the exploratory design process is also supported by the possibility to backtrack to previous states (R_2). In a similar setting, Lyra's authors observed that the absence of undo had a negative impact on users' willingness to explore alternative designs [39]. Graphies not only supports basic undo, but branching as well, enabling designers to explore several alternative designs in parallel (R_4). A new state is created when: nodes or links are added/removed from the canvas; and when visual mappings are modified. Past states are accessed from the timeline (Figure 2). Branches appear as multiple connected tracks in the timeline, as can be seen in Figure 1. Each small circle corresponds to a past state. Hovering a circle pops-up a preview thumbnail image rendering of the diagram in that state, to make it easier for users to find the appropriate one. Resuming editing from a past state automatically creates a new branch in the timeline. Loading a saved project restores the entire history, including branches.

Finally, Graphies makes it possible for users to save a set of visual mappings for later reuse with similar multivariate graphs, that have the same types of nodes and links, *i.e.*, described by the same data attributes as those involved in the mappings considered. As mentioned earlier (Section 3), the request for such a feature came from the data analysts we worked with. They sometimes have to present answers to similar questions from different clients, and they would like to be able to reuse one of their prior visualizations as a starting point. Being able to reuse visual mappings in this way brings an advantage in terms of task efficiency that can be compared with template-based approaches. But as opposed to template-based approaches, we do not expect this feature to negatively impact creativity. Indeed, Graphies does not feature any predefined mapping. It merely stores *user-defined* ones, that result from the designer's creative process. Furthermore, as detailed above, our mapping evaluation strategy does not enforce mappings in later steps. Mappings to be reused thus only aim at bootstraping the design process. Users are then free to customize their diagram at will by editing or creating new visual mappings, and by making individual adjustments to its elements.

4.5 Exporting Visual Designs

Diagrams designed with Graphies can be exported as SVG files, but more elaborate export functions relevant to the use of node-link diagrams for communication purposes have also been implemented (R_5). Designers can export selected keyframes (stages) as an image gallery, with the possibility to add textual annotations to each image in order to tell a story [4, 25].

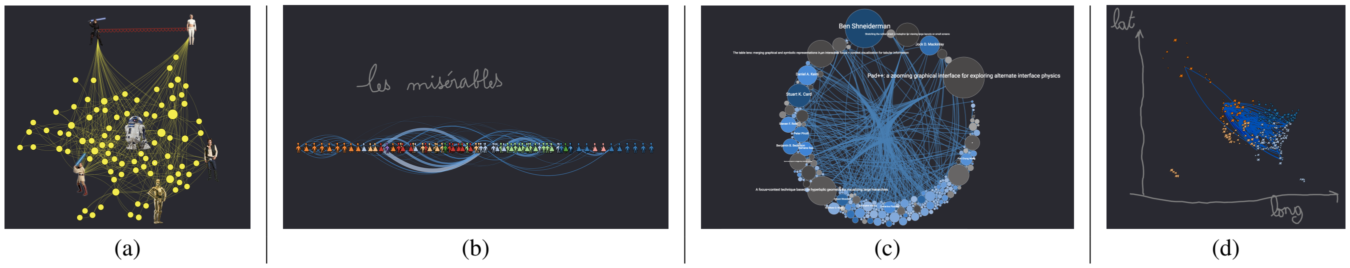

Fig. 5. Sample node-link diagrams designed with Graphies: (a) Star Wars character co-occurrences; (b) character co-occurrences in Victor Hugo's Les Misérables; (c) subset of SIGCHI publications and authors with keyword *"Information Visualization"*; and (d) flights in the USA. Annotations are used in (b) to add a title, and in (d) to show the axes used for this attribute-driven layout.

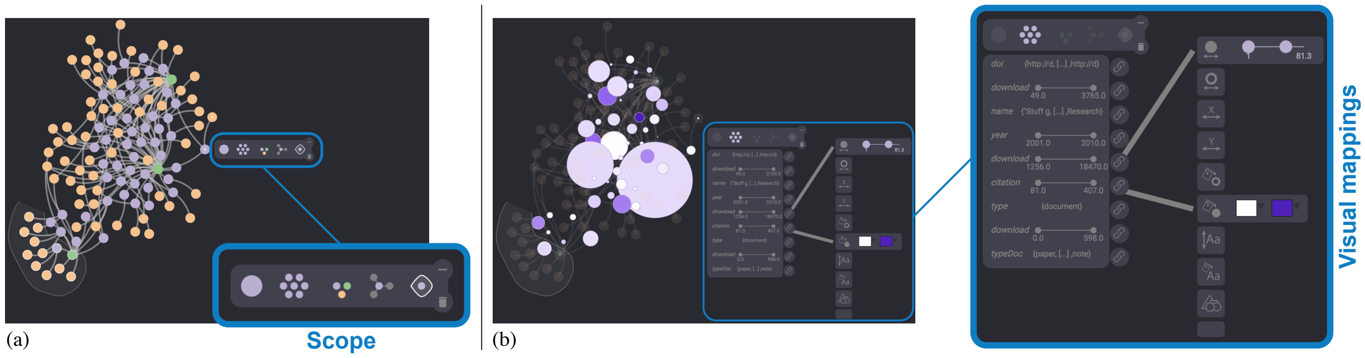

Fig. 6. Visual mapping widget. (a) A long press on a node pops up a widget to set the scope of the mappings to be created. (b) Creating mappings by connecting attributes to visual variables, and adjusting ranges, all with immediate visual feedback.

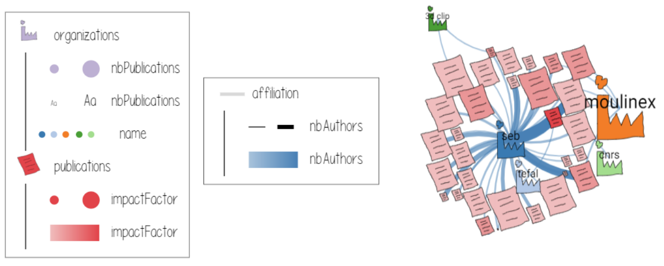

Fig. 7. Legend generated automatically by Graphies for nodes and links.

The evolution of a graph over time [5] can also be animated and exported as a video clip (MPEG-4). Here again, designers select stages from the timeline, that they use to populate the track of a simple video editor. These states, which can be reordered, define the keyframes of the animation. Graphies then renders the video clip by smoothly interpolating between those keyframes.

As discussed in Section 3, such animations can be useful when telling stories about the data: incrementally bringing data to make a point step-by-step, showing complementary perspectives on the data, showing the actual evolution of a network over time, or even documenting the incremental construction of the diagram. To further support the use of these diagrams for communication purposes, Graphies enables designers to add freeform annotations to the canvas, as illustrated in Figures 5-b & 5-d.

5 User Study

The interaction model proposed in Graphies differing significantly from that of other network visualization tools, we conducted a *first-use study* [19] to gather empirical observations about: 1) users' ability to reproduce existing expressive visualizations using Graphies; and 2) how effective the tool is at supporting the creation of expressive designs.

5.1 Procedure

Participants start the experiment by following a tutorial ($\leq 30mn$) introducing Graphies using a dataset about interactions between Star Wars characters (that dataset was not used afterwards). The operator follows a script, demonstrating a series of features in the same manner to all participants. Each time a new feature is introduced, he asks participants to use it on a different example to make sure they have understood how to use it. The list of features is: *add all nodes of a given type; add only nodes that match an attribute value filter; add all links of a given type to a specific node; then to a selection of nodes; change the visual variables of a set of elements; create a visual mapping; use the selection menu to remove some nodes and links; bundle or fan a selection of links; add keyframes to the video editor and play the resulting video; pan, zoom and reset the view; make some annotations*.

Participants then have to reproduce each of the three visualizations in Figure 8, which use visual presentation techniques from the recent literature. These are *re-creation* or *reproduction* tasks, conceptually similar to those used to evaluate many recent visualization authoring tools, *e.g.*, [25–27, 35, 39, 54]. Participants are presented with a printed image of each visualization, accompanied by a short text listing the main steps (what attribute to show, what visual encoding to use, *etc.*), in order to avoid ambiguities. The three tasks are always presented in the same order.

The diagram in task T_1 shows documents about four product categories, as well as their authors. The two node types use different sketch-based glyphs. The size of author nodes indicates how many documents they have authored.

Task T_2 is about reproducing a co-authorship network. The network consists of authors and publications. Fill color encodes node type. Size encodes the number of articles published (author nodes), or the number of co-authors (publication nodes). Emphasis is put on central actor Ben Shneiderman by 1) using his picture to decorate his node (as in, *e.g.*, Vizster [20]); and 2) forcing all publication nodes on the left and all co-author nodes on the right, bundling edges interactively [36] on each side.

While tasks T_1 and T_2 use a topology-driven layout, T_3 uses an attribute-driven layout inspired by PivotGraph [53]. Participants have to

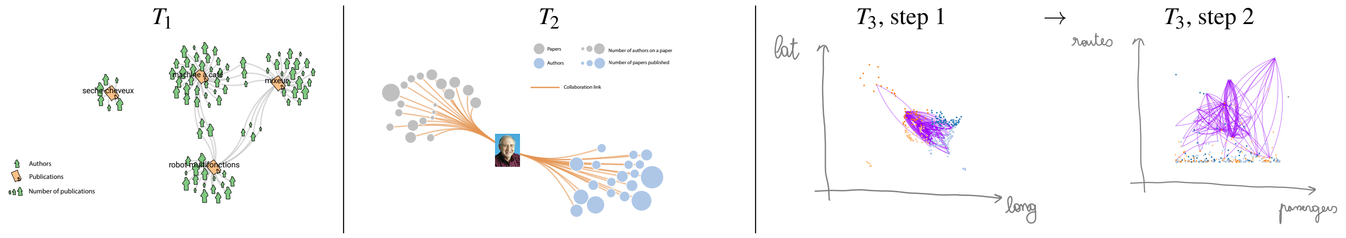

Fig. 8. The three target visualizations that participants had to reproduce during the experiment.

create an animation between two views on a US air traffic network. Step 1: airport nodes are laid out according to their latitude and longitude. Links show routes between south-east and north-west airports only. Stroke-width encodes the number of carriers on a route. Step 2: the same airport nodes are laid out in a scatterplot-like manner, where the number of transiting passengers is mapped to the x-axis, and the number of routes connecting this airport is mapped to the y-axis. In both steps, participants have to use free-form annotations to indicate the axes.

Finally, participants are asked to perform an open-ended task (T_{free}): design a visualization of their choosing about publications in the field of Information Visualization. The visualization should show the most prolific authors and their articles, emphasizing highly-visible articles. Instructions make it clear that *highly-visible* should be interpreted here as *having a large number of citations*.

5.2 Participants & Apparatus

Ten volunteers (1 female), aged 21 to 42 year-old (average 27, median 25), participated in the experiment. Three of them were undergraduate students (P2, P7, P9), three were software engineers (P4, P6, P8), three were PhD students in HCI (P0, P1, P3) and one was a PhD student in Information Visualization (P5). We conducted the experiment on a Microsoft Surface Book 2 13" (3000 × 2000 pixels, Intel Core i7 processor, 16GB RAM, Nvidia GTX 1050 graphics card).

5.3 Results

Figure 9 shows all visualizations produced by the participants, using the last frame of the animation (step 2) for T_3.[2] All participants managed to properly reproduce the target visualizations from Figure 8 *on their own*. Indeed, once the tutorial completed, the operator never gave any additional explanation about Graphies' features. He only answered the few questions participants had about how precise their reproductions had to be, telling them that the goal was to achieve a visualization that would *show the same data*, using the *same visual encodings*; but that they should not try to accurately replicate the spatial layout and link bundles, or use the exact same sketches (for nodes in T_1) and colors.

To give an indication, completion times per task were (in seconds): $353 \pm 90s$ (T_1), $498 \pm 81s$ (T_2) and $582 \pm 59s$ (T_3). But more importantly, all reproductions, regardless of the task and participant, feature the correct number of nodes and links. Given the above instructions, there are obviously variations in the local placement of elements, but the global layout strategy matches that of the templates, as do colors and other visual variables (compare Figures 8 & 9).

Participants spent $463 \pm 120s$ on the open-ended task (T_{free}) where they had to *create a design of their own*. What is striking when looking at the rightmost column in Figure 9 is the diversity of visualizations that participants created from the same dataset. While 4 participants exclusively made use of the *size* visual variable in their mappings, all others used additional channels, including position, color, transparency, and node shape (sketched glyphs). As expected, several of them also performed manual layout adjustments.

The diagrams produced by individual participants in T_{free} (rightmost column in Figure 9) demonstrate that users can be very creative when designing node-link diagrams. This diversity is an encouraging result for Graphies. Indeed, participants were able to create those visualizations in a very short amount of time. Considering that they were not professional designers,[3] the study results suggest that the approach has good potential overall.

In addition to these high-level observations, we noted interesting behaviors and gathered feedback from participants. We discuss these in the next section, providing guidelines for the development of expressive node-link diagram authoring environments.

6 Guidelines

We derive the following guidelines from a revisitation of our initial requirements in light of our empirical findings.

Seamlessly integrate all actions into the same workspace and enable users to interleave them at will. Participants followed different workflows. Some started by populating the workspace with all relevant elements, spatially arranging them before creating any visual mapping. Others quickly moved to creating visual mappings, intermingling actions to populate the canvas, arrange elements, and change their visual appearance. Flexibility in the authoring process (R_1) had been identified as an essential aspect in prior work about expressive design environments [9, 10, 27], and is confirmed here to be of key importance for node-link diagrams as well.

Support the principles of direct manipulation and provide immediate visual feedback. In particular, participants commented positively about populating the canvas by drag-and-drop from the meta-graph using a query-first strategy (Section 4.1); especially participants who had started by adding all elements of a given type before backtracking, as their diagram had become too complicated. The mapping evaluation strategy did not seem to cause confusion among participants, comforting us in our choice not to enforce the mappings at all times but on-demand only (Section 4.4). We argue, though, that this strongly depends on designers being able to perform rapid, incremental, reversible actions whose effects are visible immediately (R_2).

Provide users with effective selection mechanisms. One aspect of direct manipulation that we had not formally captured in our initial requirements is the need for designers to easily make various types of node and link selections. Selection is a central activity in an expressive design context: specifying the scope of visual mappings, making manual adjustments to node positions, link geometry, removing elements, *etc.* The selection process can be tedious, especially when working with dense, heterogeneous networks. While Graphies already supports some advanced selection mechanisms with its contextual menus, this could be taken further, for instance supporting the advanced subgraph selection and manipulation techniques by McGuffin & Jurisica [28].

Enable users to specify a wide range of visual mappings. Several study participants spontaneously commented positively about the richness of visual mappings. Six out of ten participants made use of multiple, diverse encoding channels in the open-ended task (T_{free}) including sketched shapes, confirming the need for expressiveness in terms of visual encoding channels (R_3).

[2]Study material, animations created by participants included, available online: `https://hugoromat.github.io/graphies/user_study.html`

[3]All participants had a background in Computer Science. Except for P5, none had significant experience in Information Visualization. At best, the three PhD students had followed an introductory course.

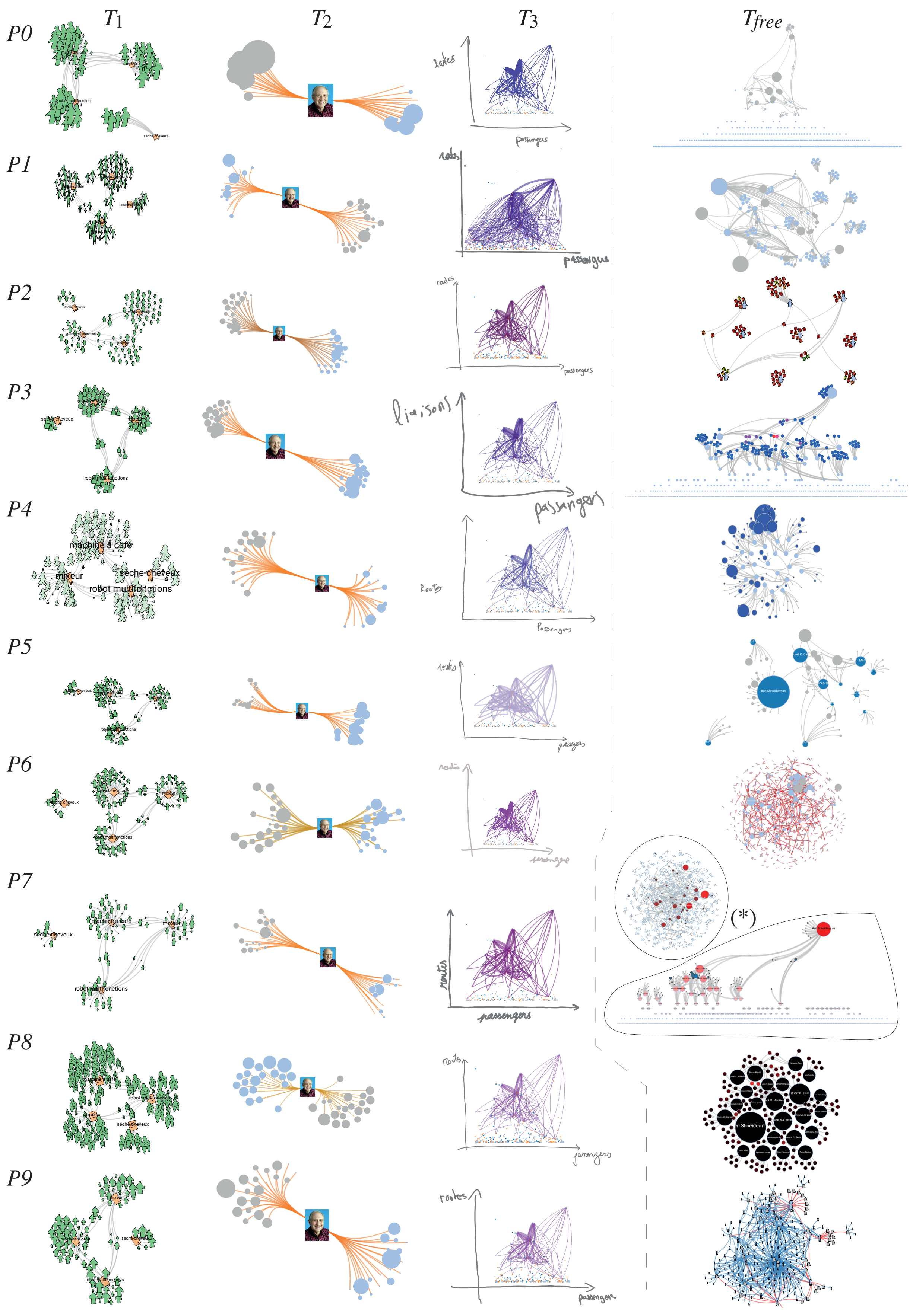

Fig. 9. Visualizations created by all ten study participants (one line per participant, zoom-in to see details). From left to right: T_1, T_2, T_3 (step 2) and T_{free}. (*) P7 asked for the possibility to make two complementary visualizations for T_{free}.

Enable users to explore multiple designs concurrently for the same multivariate network. Task T_{free} was too short to yield meaningful observations about having support for alternative design exploration. But requests from the data analysts during Graphies' development phase led to two of our initial requirements. Not only should expressive design environments support undo [39] through a history of edits, but forking at any stage in the design process as well, enabling users to keep multiple branches active (R_4). The environment should also support smooth transitions between related designs (R_5) to help compare them, as well as to enable the creation of animated node-link diagrams.

7 Conclusion and Future Work

Part of a broader effort to explore expressive approaches to the design of data visualizations, Graphies is the first to specifically focus on node-link diagram representations of multivariate networks, taking into account the high level of complexity of these data structures. With Graphies, users perform all actions in a single workspace: incrementally populating the canvas with data, specifying visual mappings and applying them on-demand, customizing individual elements by direct manipulation. Combined with a rich set of visual mappings and with the possibility to explore multiple design alternatives concurrently, this yields a flexible workflow for the expressive design of node-link diagrams. Our study confirms this, as we observed participants taking advantage of this flexibility to both streamline their actions and explore a variety of visual mappings.

Yet there are still a few limitations to consider in the design of future expressive design environments for node-link diagrams. First, Graphies keeps all visual mapping widgets close at hand in order to make users able to easily iterate on their designs (R_1) from the main canvas. While we have dedicated much effort to designing a compact widget, the fact that a new one gets instantiated each time a new visual mapping is created means that the workspace can get cluttered quickly. Users can move, minimize and even discard the widgets. Our study, however, was too short to observe whether users will actually take the time necessary to organize their workspace. A longitudinal study would be necessary to evaluate the usefulness of widget management mechanisms, that would for instance automatically minimize or rearrange them based on the user's activity.

A related problem mentioned by one participant is the potential difficulty to relate widgets to their scope. We had actually discussed this issue during the design phase. But while we can use transient highlighting to emphasize the scope of mappings one at a time (using, *e.g.*, legends - Section 4.3), we have not yet found a satisfying solution that would work for all widgets simultaneously without causing considerable clutter.

Data analysts involved in the design process also made requests for ways to quickly select data of interest. One simple request was to include a global search function to access specific elements whose name is already known, when the designer is highly familiar with the dataset. More interestingly, in the opposite context of working with a dataset that is not necessarily very familiar to the designer, the analysts asked if Graphies could suggest a subset of nodes and links to start populating the diagram based on metrics such as centrality or similarity. They indicated that this would be, again, particularly useful in the context of unfamiliar datasets. Such features, which echo Crnovrsanin *et al.*'s recommendations for network navigation [13], could also help users intimidated by a blank canvas [39].

References

[1] J. Abello, F. van Ham, and N. Krishnan. Ask-graphview: A large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669–676, 2006. doi: 10.1109/TVCG.2006.120

[2] E. Adar. Guess: A language and interface for graph exploration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pp. 791–800. ACM, 2006. doi: 10.1145/1124772.1124889

[3] D. Auber. *Tulip —A Huge Graph Visualization Framework*, pp. 105–126. Springer, 2004. doi: 10.1007/978-3-642-18638-7_5

[4] B. Bach, N. Kerracher, K. W. Hall, S. Carpendale, J. Kennedy, and N. Henry Riche. Telling stories about dynamic networks with graph comics. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '16, pp. 3670–3682. ACM, New York, NY, USA, 2016. doi: 10.1145/2858036.2858387

[5] B. Bach, E. Pietriga, and J.-D. Fekete. Graphdiaries: Animated transitions andtemporal navigation for dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):740–754, 2014. doi: 10.1109/TVCG.2013.254

[6] J. Barnes and P. Hut. A hierarchical o(n log n) force-calculation algorithm. *Nature*, 324:446 EP –, 1986.

[7] M. Bastian, S. Heymann, M. Jacomy, et al. Gephi: an open source software for exploring and manipulating networks. In *3rd International Conference on Weblogs and Social Media*, ICWSM, pp. 361–362. AAAI, 2009. doi: 10.13140/2.1.1341.1520

[8] A. Bezerianos, F. Chevalier, P. Dragicevic, N. Elmqvist, and J.-D. Fekete. Graphdice: A system for exploring multivariate social networks. In *Proceedings of the Eurographics / IEEE - VGTC Conference on Visualization*, EuroVis'10, pp. 863–872. Eurographs Association, 2010. doi: 10.1111/j.1467-8659.2009.01687.x

[9] A. Bigelow, S. Drucker, D. Fisher, and M. Meyer. Reflections on how designers design with data. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '14, pp. 17–24. ACM, 2014. doi: 10.1145/2598153.2598175

[10] A. Bigelow, S. Drucker, D. Fisher, and M. Meyer. Iterating between tools to create and edit visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):481–490, 2017. doi: 10.1109/TVCG.2016.2598609

[11] M. Bostock, V. Ogievetsky, and J. Heer. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011. doi: 10.1109/TVCG.2011.185

[12] L. Bounegru, T. Venturini, J. Gray, and M. Jacomy. Narrating networks: Exploring the affordances of networks as storytelling devices in journalism. *Digital Journalism*, 5(6):699–730, 2017. doi: 10.1080/21670811.2016.1186497

[13] T. Crnovrsanin, I. Liao, Y. Wuy, and K.-L. Ma. Visual recommendations for network navigation. In *Proceedings of the Eurographics / IEEE - VGTC Conference on Visualization*, EuroVis'11, pp. 1081–1090, 2011. doi: 10.1111/j.1467-8659.2011.01957.x

[14] C. Dunne, N. Henry Riche, B. Lee, R. Metoyer, and G. Robertson. Graphtrail: Analyzing large multivariate, heterogeneous networks while supporting exploration history. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pp. 1663–1672. ACM, 2012. doi: 10.1145/2207676.2208293

[15] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull. Graphviz and Dynagraph – static and dynamic graph drawing tools. In *Graph Drawing Software*, pp. 127–148. Springer-Verlag, 2003. doi: 10.1007/978-3-642-18638-7_6

[16] L. Grammel, C. Bennett, M. Tory, and M.-A. Storey. A Survey of Visualization Construction User Interfaces. In M. Hlawitschka and T. Weinkauf, eds., *EuroVis - Short Papers*, pp. 19–23. The Eurographics Association, 2013. doi: 10.2312/PE.EuroVisShort.EuroVisShort2013.019-023

[17] L. Grammel, M. Tory, and M.-A. Storey. How information visualization novices construct visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):943–952, 2010. doi: 10.1109/TVCG.2010.164

[18] T. R. G. Green, M. Petre, et al. Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *Journal of visual languages and computing*, 7(2):131–174, 1996. doi: 10.1006/jvlc.1996.0009

[19] B. Hartmann, S. R. Klemmer, M. Bernstein, L. Abdulla, B. Burr, A. Robinson-Mosher, and J. Gee. Reflective physical prototyping through integrated design, test, and analysis. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, UIST '06, pp. 299–308. ACM, 2006. doi: 10.1145/1166253.1166300

[20] J. Heer and D. Boyd. Vizster: Visualizing online social networks. In *Proceedings of the Proceedings of the IEEE Symposium on Information Visualization*, InfoVis '05, pp. 32–39. IEEE, 2005. doi: 10.1109/INFOVIS.2005.39

[21] N. Henry, J.-D. Fekete, and M. McGuffin. Nodetrix: a hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1302–1309, 2007. doi: 10.1109/TVCG.2007.70582

[22] S. Huron, S. Carpendale, A. Thudt, A. Tang, and M. Mauerer. Constructive visualization. In *Proceedings of the Conference on Designing Interac-*

tive Systems, DIS '14, pp. 433–442. ACM, 2014. doi: 10.1145/2598510.2598566

[23] I. Jusufi, Y. Dingjie, and A. Kerren. The network lens: Interactive exploration of multivariate networks using visual filtering. In *International Conference Information Visualisation (IV)*, pp. 35–42. IEEE, 2010. doi: 10.1109/IV.2010.15

[24] H. Kang, C. Plaisant, B. Lee, and B. B. Bederson. Netlens: Iterative exploration of content-actor network data. *Information Visualization*, 6(1):18–31, 2007. doi: 10.1057/palgrave.ivs.9500143

[25] N. W. Kim, N. Henry Riche, B. Bach, G. A. Xu, M. Brehmer, K. Hinckley, M. Pahud, H. Xia, M. McGuffin, and H. Pfister. DataToon: Drawing Dynamic Network Comics With Pen + Touch Interaction. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '19, pp. 1–12. ACM, 2019. doi: 10.1145/3290605.3300335

[26] N. W. Kim, E. Schweickart, Z. Liu, M. Dontcheva, W. Li, J. Popovic, and H. Pfister. Data-driven guides: Supporting expressive design for information graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):491–500, 2017. doi: 10.1109/TVCG.2016.2598620

[27] Z. Liu, J. Thompson, A. Wilson, M. Dontcheva, J. Delorey, S. Grigg, B. Kerr, and J. Stasko. Data Illustrator: Augmenting Vector Design Tools with Lazy Data Binding for Expressive Visualization Authoring. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '18, pp. 123:1–123:13. ACM, 2018. doi: 10.1145/3173574.3173697

[28] M. J. McGuffin and I. Jurisica. Interaction techniques for selecting and manipulating subgraphs in network visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):937–944, 2009. doi: 10.1109/TVCG.2009.151

[29] G. G. Méndez, U. Hinrichs, and M. A. Nacenta. Bottom-up vs. top-down: Trade-offs in efficiency, understanding, freedom and creativity with infovis tools. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '17, pp. 841–852. ACM, 2017. doi: 10.1145/3025453.3025942

[30] G. G. Méndez, M. A. Nacenta, and S. Vandenheste. iVoLVER: Interactive Visual Language for Visualization Extraction and Reconstruction. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '16, pp. 4073–4085. ACM, 2016. doi: 10.1145/2858036.2858435

[31] C. Nobre, M. Meyer, M. Streit, and A. Lex. The state of the art in visualizing multivariate networks. *Computer Graphics Forum*, 38(3):807–832, 2019. doi: 10.1111/cgf.13728

[32] W. D. Nooy, A. Mrvar, and V. Batagelj. *Exploratory Social Network Analysis with Pajek*. Cambridge University Press, 2011.

[33] R. Pienta, F. Hohman, A. Endert, A. Tamersoy, K. Roundy, C. Gates, S. Navathe, and D. H. Chau. Vigor: Interactive visual exploration of graph query results. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):215–225, 2018. doi: 10.1109/TVCG.2017.2744898

[34] D. Ren, T. Höllerer, and X. Yuan. iVisDesigner: Expressive Interactive Design of Information Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2092–2101, 2014. doi: 10.1109/TVCG.2014.2346291

[35] D. Ren, B. Lee, and M. Brehmer. Charticulator: Interactive construction of bespoke chart layouts. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):789–799, 2019. doi: 10.1109/TVCG.2018.2865158

[36] N. H. Riche, T. Dwyer, B. Lee, and S. Carpendale. Exploring the design space of interactive link curvature in network diagrams. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pp. 506–513. ACM, 2012. doi: 10.1145/2254556.2254652

[37] H. Romat, C. Appert, B. Bach, N. Henry-Riche, and E. Pietriga. Animated edge textures in node-link diagrams: A design space and initial evaluation. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '18, pp. 187:1–187:13. ACM, 2018. doi: 10.1145/3173574.3173761

[38] H. Romat, D. Lebout, E. Pietriga, and C. Appert. Influence of color and size of particles on their perceived speed in node-link diagrams. In *Proceedings of the International Conference on Human-computer Interaction*, INTERACT'19. Springer, 2019.

[39] A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. *Computer Graphics Forum*, 33(3):351–360, 2014. doi: 10.1111/cgf.12391

[40] Z. Shen, K.-L. Ma, and T. Eliassi-Rad. Visual analysis of large heterogeneous social networks by semantic and structural abstraction. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1427–1439, 2006. doi: 10.1109/TVCG.2006.107

[41] L. Shi, Q. Liao, H. Tong, Y. Hu, Y. Zhao, and C. Lin. Hierarchical focus+context heterogeneous network visualization. In *Pacific Visualization Symposium*, pp. 89–96. IEEE, 2014. doi: 10.1109/PacificVis.2014.44

[42] B. Shneiderman. Direct manipulation: A step beyond programming languages. *IEEE Computer*, 16(8):57–69, 1983. doi: 10.1109/MC.1983.1654471

[43] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):733–740, 2006. doi: 10.1109/TVCG.2006.166

[44] B. Shneiderman and C. Dunne. Interactive network exploration to derive insights: Filtering, clustering, grouping, and simplification. In *Proceedings of the Conference on Graph Drawing*, GD'12, pp. 2–18. Springer-Verlag, 2013. doi: 10.1007/978-3-642-36763-2_2

[45] M. A. Smith, B. Shneiderman, N. Milic-F rayling, E. Mendes Rodrigues, V. Barash, C. Dunne, T. Capone, A. Perer, and E. Gleave. Analyzing (Social Media) Networks with NodeXL. In *Proceedings of the International Conference on Communities and Technologies*, pp. 255–264. ACM, 2009. doi: 10.1145/1556460.1556497

[46] A. Spritzer, J. Boy, P. Dragicevic, J.-D. Fekete, and C. M. Dal Sasso Freitas. Towards a smooth design process for static communicative node-link diagrams. *Computer Graphics Forum*, 34(3):461–470, 2015. doi: 10.1111/cgf.12658

[47] Tableau. Tableau Desktop. `https://www.tableau.com`, 2019. [online; accessed 2019-08-26].

[48] S. van den Elzen and J. van Wijk. Multivariate network exploration and presentation: From detail to overview via selections and aggregations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2310–2319, 2014. doi: 10.1109/TVCG.2014.2346441

[49] F. van Ham and A. Perer. "search, show context, expand on demand": Supporting large graph exploration with degree-of-interest. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):953–960, 2009. doi: 10.1109/TVCG.2009.108

[50] F. Viegas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon. ManyEyes: a Site for Visualization at Internet Scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1121–1128, 2007. doi: 10.1109/TVCG.2007.70577

[51] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. van Wijk, J.-D. Fekete, and D. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011. doi: 10.1111/j.1467-8659.2011.01898.x

[52] C. Ware, H. Purchase, L. Colpoys, and M. McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002. doi: 10.1057/palgrave.ivs.9500013

[53] M. Wattenberg. Visual exploration of multivariate graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pp. 811–819. ACM, 2006. doi: 10.1145/1124772.1124891

[54] H. Xia, N. Henry Riche, F. Chevalier, B. De Araujo, and D. Wigdor. DataInk: Direct and Creative Data-Oriented Drawing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '18, pp. 223:1–223:13. ACM, 2018. doi: 10.1145/3173574.3173797