Eurographics Conference on Visualization (EuroVis) 2015
H. Carr, K.-L. Ma, and G. Santucci
(Guest Editors)

Volume 34 (2015), Number 3

Towards a smooth design process for static communicative node-link diagrams

Andre Suslik Spritzer[1], Jeremy Boy[2,3], Pierre Dragicevic[2], Jean-Daniel Fekete[2] and Carla Maria Dal Sasso Freitas[1]

[1]Instituto de Informática, Universidade Federal do Rio Grande do Sul, Brazil
[2]INRIA, France
[3]Telecom ParisTech, France

Abstract

Node-link infographics are visually very rich and can communicate messages effectively, but can be very difficult to create, often involving a painstaking and artisanal process. In this paper we present an investigation of node-link visualizations for communication and how to better support their creation. We begin by breaking down these images into their basic elements and analyzing how they are created. We then present a set of techniques aimed at improving the creation workflow by bringing more flexibility and power to users, letting them manipulate all aspects of a node-link diagram (layout, visual attributes, etc.) while taking into account the context in which it will appear. These techniques were implemented in a proof-of-concept prototype called GraphCoiffure, which was designed as an intermediary step between graph drawing/editing software and image authoring applications. We describe how GraphCoiffure improves the workflow and illustrate its benefits through practical examples.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical user interfaces (GUI)

1. Introduction

Static node-link diagrams are an effective way of communicating information, which researchers use to present their findings, journalists to illustrate their stories, and artists to express themselves. They can appear on scientific posters, or in textbooks, papers, newspapers, and magazines; and they come in many very distinct styles. However, creating such communicative node-link diagrams is not an easy task. The contexts in which they appear impose many constraints, ranging from stylistic choices (i.e., the image's "look-and-feel") to medium limitations (e.g., size, color palette, etc.); and whoever creates them must have a clear understanding of the underlying data, as well as of the message that the diagram should convey. This requires skills that a single person may not necessarily possess.

Currently available approaches for creating communicative node-link diagrams are problematic as they tend to be too labor-intensive, time-consuming, or inflexible. For example, creating a node-link diagram from scratch in a graphics editor can be painstaking and is only feasible for graphs of very limited size—such software only deal with pixels and/or vectors, and are agnostic to the data. As an alternative, analysis-centric software can be used, but these rarely allow the necessary flexibility to conform to the aesthetic requirements of communication-centered contexts. Similarly, graph editors and graph drawing applications can produce nice-looking visualizations, but they too lack flexibility for visual encoding and layout, and are truly practical only if the visualizations can be used "as is." A final alternative is to code visualizations, but this is impractical, time-consuming, and requires programming skills, which not everyone has.

In this paper, we lay the groundwork for techniques and applications aimed at supporting the creation of custom static communicative node-link diagrams, with an emphasis on tools for interactive layout manipulation. Our contributions are as follows:

- a breakdown of static node-link diagrams into their basic elements from a graphic design perspective, based on a thorough study of representative example images;
- an analysis of the workflow of designers/storytellers who create these images;
- a set of high-level tasks that may need to be performed in order to create the diagrams;
- a set of techniques for interactive layout manipulation that

take into account the graph and its associated data, as well as the visualization's semantics and context of use;
- a stylesheet-based technique for the definition of a node-link diagram's visual attributes;
- a discussion of how the proposed techniques fit into and enhance the workflow of creators of node-link visualizations for communication;

Moreover, we describe two usage scenarios showing in practice how our approach can improve the workflow and give users more possibilities.

2. Related work

2.1. Communicative node-link diagrams

There are many network visualizations depicting varied topics such as the posters sold by Pop Chart Labs [Pop14] and curated by Visual Complexity [Lim13, Lim14]. All of these visualizations have a common denominator, which is their focus on communication and the attempt to make the graphs not only meaningful, but also beautiful.

To better understand how such communicative node-link diagrams are created, we interviewed sociologist Dana Diminescu, who leads the eDiasporas project, which studies specific interactions among "migrant sites" (or *e-diasporas*)—websites created or managed by migrants and/or significantly related to migration. The project involves a series of infographic booklet/poster hybrids [Dim12b] and their web counterparts [Dim12a].

Dana and her team first created the diagrams by loading the graphs they had built into graph editor Gephi, where they applied a force-directed layout and customized some visual attributes, making edges curved and mapping degrees to node radii and tags to colors. The diagrams were then exported as SVGs and touched up on Adobe Illustrator (without changing the layout) by independent graphic designers, who finally integrated them into the posters.

The end-results looked good (Figs. 1 and 2), but Dana and her team felt somewhat frustrated. The chosen layout algorithm caused some problems for the print version due to the limited space available for each graph: some nodes ended up too far from the rest of the graph and larger graphs often needed to be reduced or cropped to fit into their areas. Some layouts were also very dense, making it difficult for viewers to understand the structure of the graphs. More problems came from the impossibility of planning graph layouts for their specific page layout: descriptive text often had to be placed over parts of the diagrams and there was frequently either too little or too much whitespace.

Beyond these specific technical issues, it seemed the main frustration came from a difference of perspective and an inability of going back and forth between design stages. The graphic designers had most likely focused on more global

Figure 1: Close-up of an e-diaspora graph, as seen on the print version of Dana Diminescu's e-Diasporas project [Dim12b].

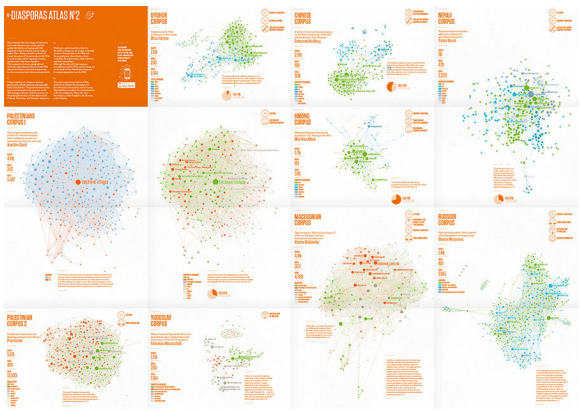


Figure 2: A page of the print version of Dana Diminescu's e-Diasporas project [Dim12b]. The page is organized as a matrix of e-diaspora graphs.

visual communication aspects, while the sociologists had mainly focused on the legibility of each individual graph.

According to Baur [Bau13], a graphic designer tries to solve problems of identification, orientation, and information, and, to this end, uses the design of visual elements just as much as any other form of creative expression that is necessary for the production of a coherent presentation. Here, **identification** concerns the more or less immediate comprehension of the major function or purpose a given object (in this case, the printed document was supposed to communicate information about social sciences); **orientation** concerns the signs on the surface of that object that communicate how the object works (in this case, the way in which the document should be read); and **information** concerns the content itself, i.e., what is being communicated (in this case, the "information" conveyed by the graphs and the text). Put simply, these three levels could be respectively translated into the questions: "What is this?", "How does it work?", and "What does it mean?" This reveals that, from a graphic

design perspective, there is a clear hierarchy among these levels, prioritizing identification and orientation over information. This conflicts directly with the hierarchy considered by researchers, which emphasizes information. In the case of e-diasporas, this clash of perspectives was exacerbated by the fact that once the work was done, it was impossible to go back, since the cost would have been too great , as the diagrams would have had to be remade almost from scratch.

In this paper, we focus on bridging these different perspectives. Our main motivation is to develop a tool that will help researchers, or any other "content providers," modify a graph at any moment to bring forward the information aspect of the communication, while allowing graphic designers to bring forward the identification and orientation aspects of the final document.

2.2. Graph editors and graph drawing software

Gephi [BHJ09] and yEd [yEd14] are free graph editors that let users create node-link diagrams. Users can apply a layout algorithm and change visual attributes such as node and edge color and edge type (e.g., curved or straight). Gephi allows for limited layout modification by letting users create metanodes from node selections, but it does not have much support for different node shapes and edge styles (e.g., dotted, dashed, different arrowheads, etc.). yEd, on the other hand, allows for more customization of the visual attributes. Both let users encode data such as computed metrics and node/edge data attributes into visual attributes.

Cytoscape [SMO*03] and NodeXL [SMFS*10] are open-source platforms for network analysis. They let users customize many visual attributes and support mapping data attributes into visual attributes. Distinctive features of Cytoscape are its plug-in architecture and its support for more layout options. NodeXL, in turn, is interesting for being a Microsoft Excel template that also lets users create diagrams from social networking platforms like Twitter and YouTube. As for layout manipulation, Cytoscape supports nested graphs while NodeXL supports metanodes and allows connected components to be separated by size.

Tulip [Aub04] is a framework for graph visualization and analysis. Based on a plug-in architecture, it was designed to support developers in creating new tools and also includes a GUI that provides access to its already implemented set of metrics, algorithms, and visualization tools (including several layout algorithms). Users can modify the visual encoding by mapping metrics to color and size, choosing among predefined node shapes, and making the edge color and thickness be interpolated from the nodes. Layouts can be modified by manually moving nodes, editing edge bends, and creating metanodes.

Graphviz [GN00] is a software package and library for graph drawing, featuring many layout algorithms (including the flagship *dot*) and giving users many choices in how nodes and edges are drawn (i.e., shape, color, style, etc.). It is largely based on DOT, a graph description language, and is run from the command line, outputting images and thus not supporting interactive layout manipulation. The graph, its layout, and visual style are all defined in the same DOT script, sometimes compromising reuse of a diagram's style.

Also worth a mention is the web-based Visual Investigative Scenarios, or VIS (`https://vis.occrp.org`), which lets journalists create infographics of business and criminal networks. It is effective within its journalistic context, but far less visually flexible than the others.

One common drawback of all of these tools is that none provides much support for layout manipulation. After choosing a layout algorithm and setting its parameters, there is not much users can do beyond moving individual nodes or grouping nodes together as metanodes. These tools also provide a very limited degree of visual customization compared to specialized graphics editors like Adobe Illustrator or Adobe Photoshop, which give users complete control of the images through many tools that range from image processing algorithms to tools for interactive drawing and image manipulation. While visually very powerful, though, these editors are not ideal for creating visualizations, as they only process pixels and vectors, providing no link to the underlying data, with all mappings having to be made by hand.

2.3. Interactive layout manipulation

Although graph drawing techniques like those described by Tamassia [Tam14] can be used to generate high-quality node-link layouts from graph topologies, this is not always enough. Some users may feel the need to manipulate layouts to make graphs more meaningful or understandable.

Some of the techniques that provide more substantial layout manipulation focus on the repositioning of nodes. McGuffin and Jurisica's approach lets users modify a node-link diagram by applying a chosen layout algorithm on a selected subset of nodes [MJ09]. In contrast, MagnetViz [SF12] allows users to reorganize the entire diagram through virtual magnets that attract nodes.

Other techniques focus on changing how edges are drawn. Riche et al. [RDLC12] have identified six dimensions for such techniques based on characteristics of user interaction. They illustrate these dimensions by describing four families of techniques, namely, interactive bundling, interactive edge fanning, edge magnets, and interactive edge legends. Other techniques that let users modify edges include EdgeLens [WCG03], EdgePlucking [WC07], and Schmidt et al.'s techniques for multi-touch surfaces [SNDC10].

All the techniques mentioned above were designed with a focus on the exploration of graphs. While several are able to export static images, they were not explicitly designed for the creation of stand-alone visualizations and do not provide enough flexibility for customizing graphs' visual attributes.

3. Elements of static communicative node-link diagrams

To better understand the components of communicative node-link diagrams, we surveyed 203 diagrams, sorting them into categories such as journalistic infographics, fun infographics, works of art, hand-drawn, etc [SBD*14]. Despite the graphs' variety, we were able to identify in them six distinct but interconnected components which we relate to the different levels of graphic design (Sect. 2.1).

1. **Information**
   The **data** is the graph that is being visualized, including its topology and associated data attributes.
   The **message** is what the diagram intends to communicate. It can be factual (i.e., stick to the data), higher-level (i.e., report non-trivial insights), or even go beyond the data, ignoring it altogether (e.g., for use in art and design) or being purposely inconsistent with it (e.g., to misinform the audience about the data).
2. **Orientation**
   The **layout** of a node-link diagram is the arrangement of nodes in space and the routing of the edges (i.e., the path of the edge from one node to another).
   Presentation aids are accessory visual elements that enhance the diagram's communicative power (e.g., annotations, legends, grids, convex hulls, etc).
3. **Identification**
   The **visual attributes** define the appearance of the elements of the diagram (e.g., colors, node shapes, line styles, label fonts, etc.). Along with the layout, they are what gives a diagram a distinct style.
   The **communication context** is how the diagram will be used. This includes considerations such as the medium in which it will appear (e.g., book, billboard, t-shirt, website, etc.), its purpose (e.g., an illustration on a news article, a whole-page infographic poster, a decorative work of art, etc.), and its intended audience (e.g., scientists, general public, children, etc.).

There may be intersections between these components, as communicative node-link diagrams are dependent on both the message and the communication context, which in turn are dependent on the layout and the visual attributes. For example, diagrams meant for print may be subject to restrictions of space and color while having to conform with aesthetic requirements meant to give them a specific "look-and-feel" (e.g., to make it congruent with the other elements of a page or to give it a desired visual identity).

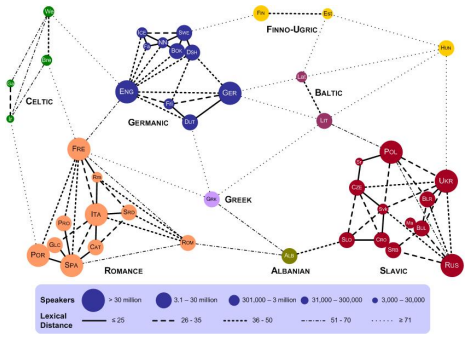


Figure 3: Teresa Elms' visualization of the lexical distance between the languages of Europe [Elm08].

To illustrate these components, let us consider Teresa Elms' diagram of the lexical distance among European languages [Elm08] (Fig. 3). This is a typical diagram that can be produced with the graph editors described in Sect. 2.2. While quite effective on the information and orientation levels, it clearly fails on the interpretation level. Indeed, data and message are made quite clear, as nodes represent languages, which are grouped into visually distinct "families," and edges show how similar languages are to one another. The layout allows for simple visual exploration of nodes and links and labels and a legend provide clear presentation aids. However, the nature of the data is quite abstract (languages) and the visual attributes do not convey much semantic information about it. A reader who only looks at this graph for a short instant (i.e., without taking the time to read all the labels) will most probably not understand what the topic is. Furthermore, the "look and feel" of this graph is quite generic and would need some additional styling to suit a specific communication context. Finally, integrating this diagram in a specific document would require shifting all other elements in the document to fit around its overall rectangular shape, which may be challenging and sub-optimal when designing a full page layout.

Based on our analysis of how static communicative node-link diagrams are usually created (Sect. 2.1), on our study of the 203 graphs, and on our breakdown of these graphs into their essential components, we derived a set of high-level tasks that users may need to perform in order to create such diagrams. Some of these tasks can be performed with image authoring programs and graph/diagram editors, while others are unique to static communicative node-link diagrams and are not directly supported by these programs. The tasks are:

- **t1:** Apply a graph layout algorithm on the diagram.
- **t2:** Manipulate the diagram's layout to make it conform to the communication context and to ensure it has the intended aesthetics and adequately communicates the intended message.
- **t3:** Manipulate the diagram's visual attributes to make it conform to the communication context and to ensure it has the intended aesthetics and adequately communicates the intended message.
- **t4:** Reproduce the style of another diagram (e.g., to create a series of visualizations with the same "look-and-feel").
- **t5:** Map information components into orientation components (i.e., use the data and the message to affect both the diagram's layout and visual attributes).
- **t6:** Add presentation aids that make the diagrams easier

to read (e.g., annotations, highlights, convex hulls, visual landmarks, legends, grids, background images, etc.).

With these tasks in mind, we designed a set of techniques for creation and manipulation of static communicative node-link diagrams and implemented them as a proof-of-concept prototype named **GraphCoiffure**, which is intended to bridge the gap between graphics editors and graph drawing software.

4. GraphCoiffure

GraphCoiffure is a prototype tool for **graph beautification**, i.e., for touching up a node-link diagram to enhance its communicative power or to make it conform to desired aesthetics. Here, we use the term "aesthetics" in a broader sense than what is usual in the graph drawing community [BETT99, WPCM02, BRSG07, PPBP10]—our concern is not only to find nice-looking, easy to read positions for nodes, but also to address the entire "look-and-feel" of a diagram. This requires considering the different visual attributes that compose a diagram (e.g., node shapes, colors, label fonts, etc.).

Users can import diagrams created with graph editors and graph drawing software into GraphCoiffure and "beautify" them with the aid of: 1) a CSS-like stylesheet system, 2) tools for interactive graph layout manipulation, and 3) page layout schemas—these can help structure a layout for specific communication contexts. GraphCoiffure preserves all visual mappings, which makes it easier to perform topology, attribute, or context-based modifications. As such, we propose the following workflow:

- **Before GraphCoiffure:** 1) create/extract a graph; 2) design a page layout; and 3) design all desired node shapes in a graphics editor.
- **In GraphCoiffure:** 1) import the graph; 2) define the visual attributes; 3) import the page; 4) touch up the layout; 5) add presentation aids; and 6) export the diagram.
- **After GraphCoiffure:** 1) add finishing touches in a graphics editor; and 2) place the image on the page.

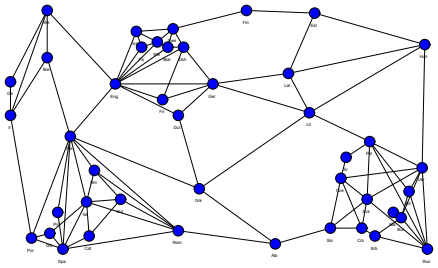
Note that these steps are not necessarily linear—users may go back-and-forth between them, or conduct them in a different order.

4.1. Touching up a graph with GraphCoiffure

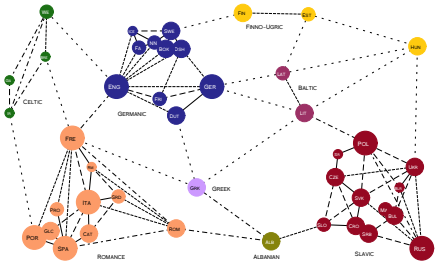
In the following subsections, we illustrate GraphCoiffure's features by describing our personal reproduction of the lexi-graph (Fig. 3). When first imported into GraphCoiffure, the graph keeps its original layout and uses GraphCoiffure's default visual attributes (as shown in Fig. 4a).

4.1.1. Using stylesheets to define visual attributes

GraphCoiffure uses a CSS-like stylesheet system to define the look of nodes, edges, and presentation aids. Users can



(a) With GraphCoiffure's default stylesheet.



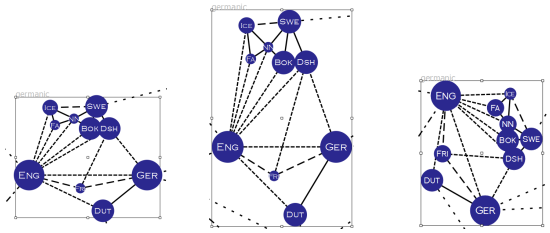
(b) With a stylesheet reproducing the lexi-graph's original style.

Figure 4: Reproducing the lexi-graph with GraphCoiffure.

customize visual attributes such as node shape, color, size, stroke patterns, fill patterns, and fonts. We chose this approach instead of a more UI-based one because of the high expressive power and reproducibility of stylesheets. These effectively help with **t3**, **t4**, and **t5**, and are immediately accessible to users familiar with CSS, like graphic designers. However, for other users, we acknowledge there may be a short learning curve.

This approach is similar to Pietriga's Graph Stylesheets (GSS)—a stylesheet language for node-link representations of RDF (Resource Description Framework) models of Semantic Web data [Pie06]. However, GraphCoiffure stylesheets are based on standard CSS, with a similar syntax and functionality; they also include several extra features, like the possibility to directly map data attributes to visual variables (e.g., number of speakers to color or size/scale), and to use topology and data-based comparison operators in a ruleset's selector to specify which nodes or edges will be affected (e.g., to make nodes of Romance language with over 7 million speakers green). The shape of nodes can also be modified with user-provided SVG images (if users don't specify a node shape, a default blue circle is used). Finally, we chose to keep the manipulation of visual attributes separate from that of the layout, as nodes and edges can be required to keep the same "look and feel" across different diagrams (e.g., in a series of related graphs, such as different "pictures" of a network evolving over time), while layouts tend to be very specific to individual graphs.

Fig. 4b shows our styled reproduction of the lexi-graph; the stylesheet we used took less than 20 minutes to write, and



(a) Original layout (b) Stretched (c) Rotated

Figure 5: Node group deformations of nodes representing Germanic languages.

contains less than 100 lines of code—despite the significant amount of information that needed to be encoded. The process was very straightforward: we created rulesets for family names, for ranges of number of speakers, for ranges of lexical distances, and for the font of the annotations. We also kept the default node shape.

4.1.2. Manipulating the layout

Besides standard click-and-drag actions, GraphCoiffure has many tools for layout manipulation that were designed to help with **t1**, **t2**, and **t5**. These tools are divided into two categories: **node group-based** and **physics-based**. Many of them are based on selections, so we will begin by explaining how this can be done in GraphCoiffure.

4.1.2.1. Selection GraphCoiffure supports three types of node selection: manual, topology-based, and attribute-based. Manual selections are the standard click-to-select, control-key + click to add selections, and click-and-drag to activate rubberband selections. Filters can also be used to target specific types of objects (e.g., only nodes, only edges, etc.).

Topology-based selections take two steps: users first manually select a group of nodes, then choose a topology-based operation from a menu (select neighbors, add neighbors to the current selection, select connected components).

Attribute-based selections use panels to select nodes or edges based on data attributes and topological properties (e.g., degree, connected component, etc.). These panels show nodes and edges in a table that can be filtered—this makes it easy, for instance, to select all languages of a given family in the lexi-graph.

4.1.2.2. Node-group-based manipulation To make localized layout manipulation easier, a **node group** can be created from a selection of nodes. Groups are represented by light gray bounding boxes around nodes, with a name (given upon creation) displayed at the top-left corner. Users can move groups by clicking and dragging, and can stretch and rotate them by clicking and dragging on manipulator handles. Stretching consists of a uniform or non-uniform scaling of the nodes' positions, while rotation rotates their positions around the center of the group's bounding box. The graph itself is treated as a special type of node group that can also be scaled as a whole (users can toggle the option to scale the whole graph or only node positions on a menu). Fig. 5 demonstrates node group deformations. Note that edge groups can also be created, but they are not used for layout modification. However, both edge and node groups can be referenced in the stylesheets described in Sect. 4.1.1.

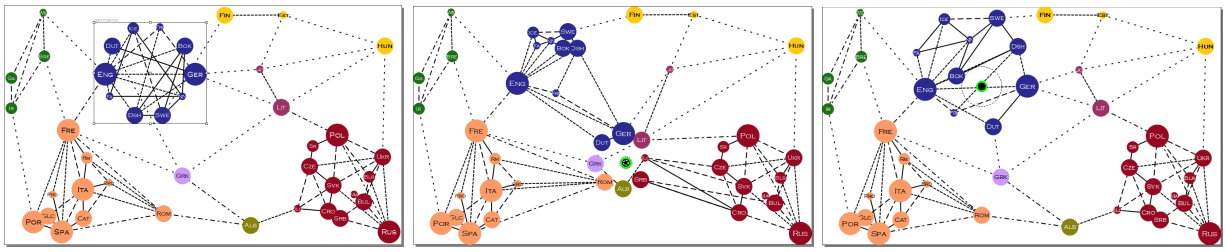
Users can also modify node groups with layout algorithms, like random, circular, spectral, and force-directed (Fig. 6a). These standard algorithms were chosen because they were readily available in the library we used to implement GraphCoiffure (networkx [HSS08]), and we considered them enough for our proof-of-concept prototype. Other algorithms can be easily added.

4.1.2.3. Physics-based tools GraphCoiffure includes an interactive force-directed layout algorithm and two physics-based tools for manipulation: a **magnet** and a **repeller**.

The interactive algorithm is a variation of Fruchterman-Reingold [FR91] that keeps a physics engine running until explicitly stopped, making the entire graph respond to changes that happen when nodes are moved. This can be useful to "model" or "smooth out" a layout, and can be applied on the entire graph or on a chosen node group. This particular algorithm was chosen due to its straightforward implementation (the prototype being merely a proof-of-concept, scalability was not a concern).

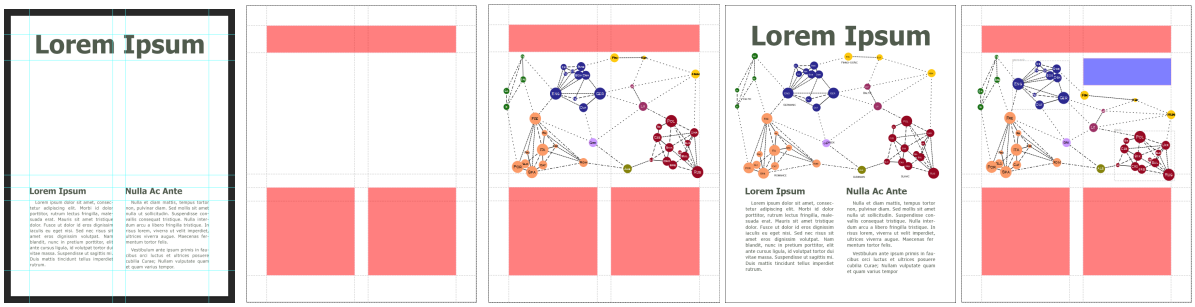
Magnets are based on the MagnetViz concept [SF12]; they apply an attraction force on a selection of nodes, which reshapes the layout (Fig. 6b). This can be used to create a desired aesthetic, to emphasize a particular subgraph, or even to clean up a diagram (e.g., disentangle subgraphs/connected components); and is particularly powerful when combined with an interactive force-directed layout. Users can place magnets anywhere on the workspace, and freely move them around, while activating or deactivating them by double-clicking on their icons. This changes the color to reflect their status (green for active, red otherwise). The strength of a magnet's attraction force can be set with menu options and keyboard shortcuts. Finally, when nodes move towards a magnet, the underlying physics engine ensures that they do not collide with each other.

Repellers move nodes away from an area delimited by the repeller's radius; they work both as a brush used to unclutter dense regions and as a tool to establish "no-go" zones in the workspace (Fig. 6c). A repeller's radius can be determined with menu options and keyboard shortcuts, and appears as a dashed black circle around its icon. Like magnets, repellers can be activated and deactivated by double-clicking.



(a) Circular layout on a subgraph (b) A magnet in action. (c) Repeller used on the Germanic languages.

Figure 6: Manipulating lexi-graph's layout



(a) The target page (in Photoshop). (b) The corresponding layout schema. (c) Scaling the lexi-graph to fit the layout. (d) The final page. (e) Variation: modifying the layout to adapt to a movable page element.

Figure 7: Using page layout schemas.

4.1.3. Presentation aids

GraphCoiffure supports annotations as presentation aids (**t6**); these are small blocks of text that can be placed anywhere on the workspace and styled with the stylesheet system. Annotations can be attached to a graph as a whole, or to specific node groups—when a group is moved or deformed, the annotation's position is updated accordingly. We use annotations in our styled reproduction of the lexi-graph (Fig. 4b).

4.1.4. Page schemas

Whatever the final format (e.g., book, poster, website, etc.), node-link diagrams often appear as one of several elements on a page. This communication context may impose restrictions on the size and shape of the graph layout. To help users plan a graph layout for a page layout (task **t2**), GraphCoiffure lets users import **page layout schemas** (Fig. 7).

A page layout schema is a design aid that can include a grid and placeholders for the page's different visual elements. The placeholders can be fixed (displayed in red on the workspace), representing elements on the page that must remain in position; or movable (in blue), representing elements that can be repositioned if necessary. Layout schemas are loaded from formatted SVG files, in which grids are made up of *line* objects with a "grid" *class* (i.e., *class*="*grid*"); and fixed and movable placeholders are *rect* objects with "fixedBlock" and "movableBlock" *classes,* respectively.

4.2. Usage scenarios

To further illustrate the use of GraphCoiffure, and the workflow that goes with it, we now present the design of three other "beautified" diagrams. The first uses the lexi-graph and focuses on making it look like the infographic of Fig. 8a. The second and third are two original diagrams we created of a network of movies.

4.2.1. Reproducing a style

The infographic in Fig. 8a was made by former company Loku; it shows different types of coffee and the different personalities of the people who drink them. In this subsection, we describe how we reproduced this diagrams style and applied it to the lexi-graph using GraphCoiffure.

As mentioned in our proposed workflow, we began outside GraphCoiffure by creating a page layout schema based on the coffee diagram's page. We also created the node shape SVGs in Adobe Illustrator, reproducing the two ways nodes are drawn. We then loaded our reproduction of the lexi-graph

in GraphCoiffure and wrote a stylesheet that used our recreated node shapes. The "personality" shape was applied to nodes with a degree greater than 6, while the "coffee type" shape was applied to the others. This stylesheet contains less than 30 lines of code and was written in about five minutes. However, while the resulting diagram (Fig. 8b) looks promising, more touching-up is needed: several nodes overlap and the overall shape of the diagram (i.e., the layout and the page) is still far from the original infographic's.

To adapt the layout, we loaded our page layout schema to serve as a reference. We then stretched and rotated the graph to make it fit the page using the node group deformation tools described in Sect. 4.1.2. After that, we created node groups for each language family and deformed them using a repeller to eliminate overlaps and to unclutter dense regions; we occasionally had to manually move individual nodes or node selections. We took special care to avoid having nodes and edges overlap areas of the page that had been assigned to other types of visual content (i.e., text or other images). This process is illustrated in Fig. 8c.

Once happy with the layout, we exported the diagram from GraphCoiffure as a regular image and loaded it in Adobe Photoshop, where we added the central image and a company logo to produce Fig. 8d.

4.2.2. Rotten Tomatoes infographics

To illustrate the flexibility of our method, we designed two versions of another graph, which counts 462 nodes and 511 edges. The data came from a subset of HetRec 2011's "MovieLens + IMDb/Rotten Tomatoes" dataset [CBK11, rg14] and presented a network of the movies of 2008 that received the most user ratings (over 50,000).

We loaded the graph in GraphCoiffure and applied an initial force-directed layout to get an idea of its general aspect. After that, we started working on the stylesheet: we encoded movies as ellipses, actors as stars, and directors as director's chairs—each accompanied with labels showing their titles and names. We mapped movies' Rotten Tomatoes audience scores to their width and their top critics score to their height; actors' average ranks (their "importance" in a movie) to their size; and actors' ranks in each of their movies to the thickness of their connecting edges. We then defined the fonts and colors.

We then worked on the graph's layout. We activated the interactive force-directed layout and we used a magnet to clean up the layout by separating smaller connected components from the largest one (with 367 nodes). As we wanted the infographic's title to appear *inside* the diagram instead of above it, we placed a repeller with a relatively large radius in a central region between the many smaller connected components and the largest one, forcing the layout to reshape itself around this region (effectively creating a hole in the layout). After the physics engine had done its work, we stopped it and used the node group tools to stretch the graph horizontally to give the infographic a landscape aspect ratio. After making some minor tweaks by manually moving some nodes to ensure their labels were readable, we exported the diagram as an SVG file and opened it in Adobe Illustrator. Finally, we added the title and a background image and made some last adjustments for the color of the objects. The final graphic is shown in Fig. 9a, a close-up of which is shown in Fig. 9b.

Next, to give a better idea of GraphCoiffure's flexibility, we created a second infographic for the same dataset, keeping the encoding of actors as stars and directors as chairs, but changing the encoding of movies so that they also take their genre into account. Movies are now shown as either tomatoes (*fresh*) or green blobs (*rotten*). Inside these icons, we added theater masks to represent the genres, and audience score was encoded as node size. Once again we separated smaller components from the large one, but we did not insert a "hole" for the logo in the center, as we decided on a different design. We also decided to keep the original proportions of the diagram. To produce the final image, we exported it and placed it over a background previously created in Illustrator. A close-up of this image is shown in Fig 9c and the full version is available on GraphCoiffure's graph collection website [SBD*14].

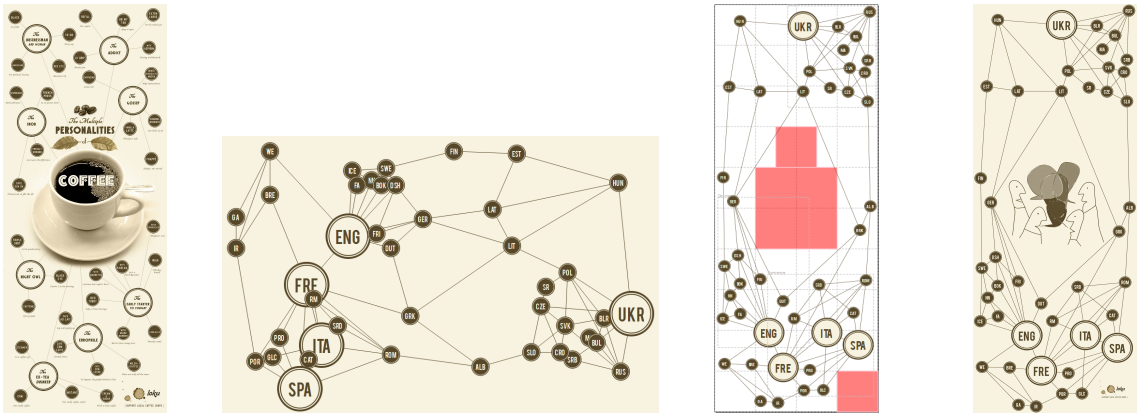
4.3. Implementation details

GraphCoiffure was developed in Python 3.3 using the PySide bindings for the C++ Qt framework. Libraries NetworkX and TinyCSS were used for the graphs and the stylesheet system, respectively. Qt's XML classes and the pyparsing library were used for the SVG parser. The physics engine was implemented with the numpy and scipy libraries.

We did not design GraphCoiffure for scalability, so in its current state it cannot be used to generate visualizations for graphs of more than a few hundred nodes and edges. This isn't really a problem, though, as our focus was more on printed visualizations, which have a natural limit on the amount of data that can be depicted due to the finite and non-interactive nature of physical surfaces.

5. Conclusion

In this paper, we have highlighted the importance of providing means to refine automatically generated graph layouts. Our interview of Dana Diminescu emphasized the importance of considering collaborators' differences of perspectives, as her example revealed that graphic designers value visual aspects over information, while researchers value the opposite. To bridge these points of view, we have proposed GraphCoiffure as a proof-of-concept prototype that helps individual users or collaborators transition between graph editors and graphics software, allowing them to beautify graphs. Although it is already powerful, we can think of



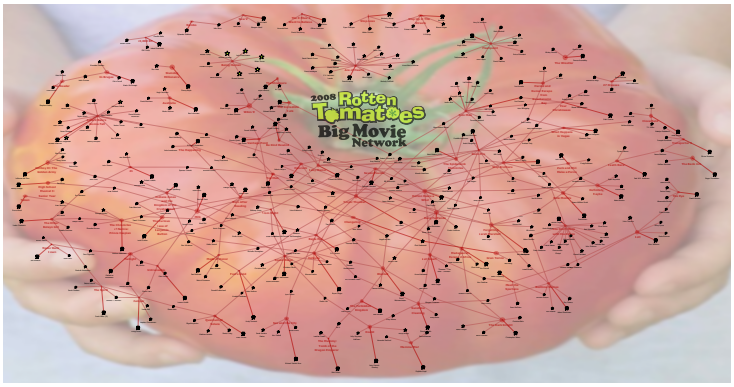
(a) The original coffee / personalities infographic.

(b) Applying a stylesheet based on the coffee infographic.

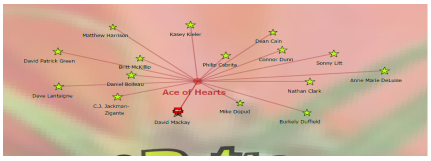
(c) Adapting the layout to the page layout schema.

(d) Re-styled lexi-graph

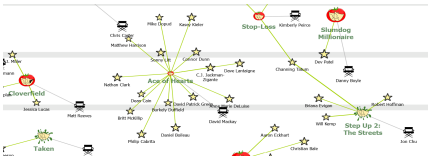
Figure 8: Making our lexi-graph reproduction look like the coffee infographic.



(a) First infographic.



(b) Close-up of the first infographic.



(c) Close-up of the second infographic.

Figure 9: Our infographics of the 2008 Rotten Tomatoes movie dataset.

several extra features that would immediately increase the practical utility of GraphCoiffure; these are support for undirected graphs, curved edges, options for edge routing, alternative edge drawings, a more sophisticated label positioning strategy, metanodes, stylesheet gradients, history management, loading graphs from formats other than GraphML (e.g., Graphviz's *dot*), and other presentation aid types (e.g, highlights, convex hulls, visual landmarks, legends, etc.).

In addition, while we have already included several layout manipulation tools, we believe there is space for the development of new techniques for deforming and reshaping layouts. This could provide users with even more power and control over the image they are composing. Another extension to our work could be to find ways to transform an image depicting a graph back into an editable diagram. Finally, it should prove interesting to fully investigate how design elements can help increase the communicative power of a visualization, as is discussed in [BF14].

Overall, we acknowledge that GraphCoiffure is merely a first step into facilitating the transition between content generators and designers when attempting to produce high quality static communicative node-link diagrams. However, our proof-of-concept prototype already helps users make quality diagrams that can be smoothly moved from screen to page.

6. Acknowledgments

This work has been partially sponsored by Brazilian funding agencies CNPq, CAPES, and FAPERGS. We also acknowledge the reviewers' comments, which helped improve the paper.

References

[Aub04] AUBER D.: Tulip: A huge graph visualization framework. In *Graph Drawing Software*, Jünger M., Mutzel P.,

(Eds.), Mathematics and Visualization. Springer Berlin Heidelberg, 2004, pp. 105–126. 3

[Bau13] BAUR R.: *Les 101 Mots du Design Graphique à l'Usage de Tous*. Archibooks, 2013. 2

[BETT99] BATTISTA G. D., EADES P., TAMASSIA R., TOLLIS I. G.: *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999. 5

[BF14] BOY J., FEKETE J.-D.: The CO2 Pollution Map: Lessons Learned from Designing a Visualization that Bridges the Gap between Visual Communication and Information Visualization. In *IEEE Conference on Information Visualization [Poster paper]* (Paris, France, Nov. 2014). To appear. 9

[BHJ09] BASTIAN M., HEYMANN S., JACOMY M.: Gephi: an open source software for exploring and manipulating networks. In *ICWSM: International AAAI Conference on Weblogs and Social Media* (2009), pp. 361–362. http://www.gephi.org. 3

[BRSG07] BENNETT C., RYALL J., SPALTEHOLZ L., GOOCH A.: The aesthetics of graph visualization. In *Proceedings of the Third Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging* (Aire-la-Ville, Switzerland, Switzerland, 2007), Computational Aesthetics'07, Eurographics Association, pp. 57–64. 5

[CBK11] CANTADOR I., BRUSILOVSKY P., KUFLIK T.: 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems* (New York, NY, USA, 2011), RecSys 2011, ACM. 8

[Dim12a] DIMINESCU D.: e-diasporas atlas. http://www.e-diasporas.fr, 2012. [accessed on June 29, 2014]. 2

[Dim12b] DIMINESCU D.: *e-Diasporas Atlas : Exploration and Cartography of Diasporas on Digital Networks*. Maison des Sciences de l'Homme, 2012. 2

[Elm08] ELMS T.: Lexical distance among the languages of europe. http://elms.wordpress.com/2008/03/04/lexical-distance-among-languages-of-europe/, 2008. [accessed on May 23, 2014]. 4

[FR91] FRUCHTERMAN T. M. J., REINGOLD E. M.: Graph drawing by force-directed placement. *Softw. Pract. Exper. 21*, 11 (1991), 1129–1164. 6

[GN00] GANSNER E. R., NORTH S. C.: An open graph visualization system and its applications to software engineering. *Software - Practice and Experience 30*, 11 (2000), 1203–1233. 3

[HSS08] HAGBERG A. A., SCHULT D. A., SWART P. J.: Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)* (Pasadena, CA USA, Aug. 2008), pp. 11–15. 6

[Lim13] LIMA M.: *Visual Complexity: Mapping Patterns of Information*. Princeton Architectural Press, 2013. 2

[Lim14] LIMA M.: Visual complexity. http://www.visualcomplexity.com, 2014. [accessed on May 23, 2014]. 2

[MJ09] MCGUFFIN M. J., JURISICA I.: Interaction techniques for selecting and manipulating subgraphs in network visualizations. *IEEE Transactions on Visualization and Computer Graphics 15* (2009), 937–944. 3

[Pie06] PIETRIGA E.: Semantic web data visualization with graph style sheets. In *Proceedings of the 2006 ACM Symposium on Software Visualization* (New York, NY, USA, 2006), SoftVis '06, ACM, pp. 177–178. 5

[Pop14] POPCHARTLAB: Pop chart lab website. http://www.popchartlab.com, 2014. [accessed on May 23, 2014]. 2

[PPBP10] PURCHASE H. C., PLIMMER B., BAKER R., PILCHER C.: Graph drawing aesthetics in user-sketched graph layouts. In *Proceedings of the Eleventh Australasian Conference on User Interface - Volume 106* (Darlinghurst, Australia, Australia, 2010), AUIC '10, Australian Computer Society, Inc., pp. 80–88. 5

[RDLC12] RICHE N. H., DWYER T., LEE B., CARPENDALE S.: Exploring the design space of interactive link curvature in network diagrams. In *Proceedings of the International Working Conference on Advanced Visual Interfaces* (New York, NY, USA, 2012), AVI '12, ACM, pp. 506–513. 3

[rg14] RESEARCH GROUP G.: Grouplens website. http://www.grouplens.org, 2014. [accessed on February 27, 2015]. 8

[SBD*14] SPRITZER A. S., BOY J., DRAGICEVIC P., FEKETE J.-D., FREITAS C. M. D. S.: Selected graphs visualizations. http://www.graphs-graphcoiffure.rhcloud.com, 2014. [accessed on March 6, 2015]. 4, 8

[SF12] SPRITZER A. S., FREITAS C. M. D. S.: Design and evaluation of magnetviz - a graph visualization tool. *IEEE Transactions on Visualization and Computer Graphics 18*, 5 (2012), 822–835. 3, 6

[SMFS*10] SMITH M. A., MILIC-FRAYLING N., SHNEIDERMAN B., RODRIGUES E. M., LESKOVEC J., DUNNE C.: Nodexl: a free and open network overview, discovery and exploration add-in for excel 2007/2010. http://nodexl.codeplex.com/ from the Social Media Research Foundation, http://www.smrfoundation.org, 2010. [accessed on June 29, 2014]. 3

[SMO*03] SHANNON P., MARKIEL A., OZIER O., BALIGA N. S., WANG J. T., RAMAGE D., AMIN N., SCHWIKOWSKI B., IDEKER T.: Cytoscape. http://www.cytoscape.org, 2003. [accessed on June 29, 2014]. 3

[SNDC10] SCHMIDT S., NACENTA M. A., DACHSELT R., CARPENDALE S.: A set of multi-touch graph interaction techniques. In *ACM International Conference on Interactive Tabletops and Surfaces* (New York, NY, USA, 2010), ITS '10, ACM, pp. 113–116. 3

[Tam14] TAMASSIA R.: *Handbook of Graph Drawing and Visualization*. CRC Press, Boca Raton, FL, 2014. 3

[WC07] WONG N., CARPENDALE S.: Supporting interactive graph exploration using edge plucking. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* (Jan. 2007), vol. 6495 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*. 3

[WCG03] WONG N., CARPENDALE S., GREENBERG S.: Edgelens: an interactive method for managing edge congestion in graphs. In *Proceedings of the Ninth annual IEEE conference on Information visualization* (Washington, DC, USA, 2003), INFOVIS'03, IEEE Computer Society, pp. 51–58. 3

[WPCM02] WARE C., PURCHASE H., COLPOYS L., MCGILL M.: Cognitive measurements of graph aesthetics. *Information Visualization 1*, 2 (June 2002), 103–110. 5

[yEd14] YED: yed. http://www.yworks.com/yed, 2014. [accessed on June 29, 2014]. 3