Graph Drawing through the Lens of a Framework for Analyzing Visualization Methods (Invited Talk, Extended Abstract)

Tamara Munzner

University of British Columbia, Department of Computer Science
Vancouver BC, Canada
tmm@cs.ubc.ca
http://www.cs.ubc.ca/~tmm

Abstract. The visualization community has drawn heavily on the algorithmic and systems-building work that has appeared with the graph drawing literature, and in turn has been a fertile source of applications. In the spirit of further promoting the effective transfer of ideas between our two communities, I will discuss a framework for analyzing the design of visualization systems. I will then analyze a range of graph drawing techniques through this lens. In the early stages of a project, this sort of analysis may benefit algorithm developers who seek to identify open problems to attack. In later project stages, it could guide algorithm developers in characterizing how newly developed layout methods connect with the tasks and goals of target users in different application domains.

1 Introduction

Visualization researchers and practitioners have long drawn on the algorithmic work conducted by the graph drawing community, and in turn have helped establish connections between that community and end users for specific application areas. Moreover, the network data that is the focus of the graph drawing community's efforts can be considered as a special case of the broader spectrum of data that is of interest in visualization, and thus its general principles are relevant.

I propose analysis of visualization techniques through *methods*; that is, an enumeration of the design space of techniques in terms of specific sets of choices. This kind of analysis supports thinking systematically about the space of possibilities. It may help a designer in the early stages of developing a new technique to identify gaps in the previous work to address. It can also be used to characterize existing work, in service of matching up which algorithms and techniques are suitable for which real-world problems. Further reading about this analysis framework can be found in an existing book chapter [13] and a forthcoming book [14]. These sources include many more references to the extensive related work that underlies this framework, which I do not directly include here.

In this talk, I begin with a distinction between four levels of visualization design, and continue with a brief discussion of abstraction for data. I introduce the principles of marks and channels, and discuss the use of space in a visualization context. I continue with further examples of analysis drawn from graph drawing, and then conclude.

2 Levels of Visualization Design

In recent work, I proposed separating the design concerns of visualization into four levels: domain problem, data and task abstraction, visual encoding and interaction technique, and algorithm, as shown in Figure 1 [10]. In that paper, I also discuss the problem of how to validate designs at each of these levels. In this talk I will emphasize techniques, which are at a level just above the algorithm level that is the focus of much of the research from the graph drawing community. I also discuss the abstraction level briefly, to provide background context.

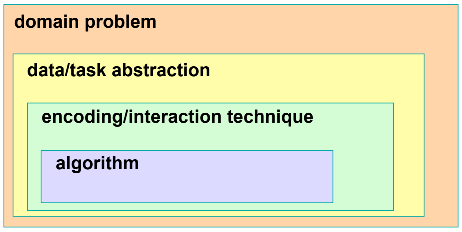

Fig. 1. Four levels of visualization design concerns [10]

My characterization here focuses on one major issue: how is space used? This question is an explicit consideration in visualization, but the motivation is not quite so obvious when considered purely from the perspective of problems that arise in graph drawing. I conjecture that the reason for this difference is that the very common cases in graph drawing, such as force-directed placement with node-link representations, or compound graphs that combine an underlying network with a hierarchy on top of it, are not trivial to analyze. My goal is to encourage more upwards characterization to map from algorithms up to techniques; that is, where algorithms are characterized in terms of the visual encoding and interaction techniques that they support.

When considering the four levels of design, another obvious route of attack is downwards from the top level of a domain problem; that is, to design a visualization system intended to solve some specific problem for a set of target users who have real data and real tasks. This sort of problem-driven work, often called *design studies* in the visualization literature has rich and interesting challenges, many of which are quite different than those that arise from technique-driven work. A detailed discussion of these issues appears elsewhere, in a recent paper on the methodology of design studies [17], and is beyond the scope of this talk.

3 Abstraction for Data

For the purposes of this framework, I will define only two basic types of data abstractions. At the dataset level, there are two major dataset types: *tables* and *networks*. In a simple table, I will call the rows *items* and the columns *attributes*. In a network, I distinguish between two kinds of items: *nodes*, and the *links* between them; either can have attributes. Obviously, the network dataset type is the focus of interest in graph drawing, but I will also present some analyses of table data as part of building up the framework. Attributes also have types. *Categorical* attributes have no implicit ordering, in contrast to *ordered* attributes; these are split into *quantitative* attributes that support full arithmetic operations such as addition or subtraction, in contrast to *ordinal*. For example, type of fruit is a categorical attribute; weight is quantitative; T-shirt size is ordinal.

The common case in visualization of complex, real-world data is that the designer will need to derive additional data beyond the original dataset. This derived data might be new attributes, or even a transformation from one dataset type to another, as with transforming a network into a table or vice versa. One example of a derived quantitative attribute computed from an original network is the Strahler number, a node-based centrality metric. Auber proposed exploiting it for fast interactive rendering of large graphs: by drawing nodes in priority order according to this attribute, a comprehensible skeleton of the network results from drawing only a small fraction of the nodes, in contrast to the poor results from drawing a random sampling [2].

4 Principles of Marks and Channels

I will introduce the idea of breaking down a visual encoding in terms of marks and channels by first considering some easy cases from statistical graphics that show tabular data: bar charts and scatterplots. These plots are straightforward to break down into *marks*, namely geometric primitives that represent items, and visual *channels* that control the appearance of marks. Marks are classified by their dimensionality: points, lines, areas, or volumes. Visual channels include spatial position, color, shape, size, orientation/tilt, and many others. A simple bar chart uses line marks, and encodes one attribute according to vertical spatial position channel. A scatterplot uses point marks, and encodes two attributes: one with the vertical spatial position channel, and one with horizontal position. A third attribute can be added to a scatterplot by encoding with the color channel, and a fourth by encoding with the size channel. The principles of marks and channels can be used to analyze more complex visual encoding techniques beyond these simple statistical graphics.

In addition to marks that represent items or nodes, marks may represent links. Link marks should implicitly convey the idea of relationships between items at a perceptual level. There are two particularly perceptually appropriate ways to do so: containment and connection. Containment uses an area mark to enclose a set of other marks within it; connection uses a line mark to directly connect two other marks together. I use the terms *connection* and *containment* for link marks, in contrast to *line* and *area* for item marks, to underscore that they communicate relationships between multiple items. A third perceptual way to indicate relationship is *proximity*, where items that are close to each other are implicitly perceived as being more related than those that are far apart. It is not possible to directly use proximity as a mark type, but in the next section I will discuss where it fits within the analysis of space use in visualization.

A crucial aspect of visual channels is that they also have implicit perceptual types, and these can and should be matched with attribute types. Some channels intrinsically convey *how much* in a way that maps well to ordered attributes, such as the spatial position along a common scale, or the length of a line mark, or the size of a point mark. Other channels convey *what* in a way that maps well to categorical attributes, such as what spatial region a mark is within, or what color a mark is, or what shape a mark is. The channels associated with the use of space have the strongest perceptual impact, leading to my choice to emphasize the spatial channels in this talk. The other channels can also be roughly ranked in terms of perceptual impact. In another talk, I discuss the underlying reasons for these rankings and a number of visualization principles that arise from them [11].

5 Using Space

I now discuss in more detail the ways to use space in the design of visual encoding and interaction techniques, emphasizing the different possible uses of spatial channels to control the appearance of marks.

I distinguish between five ways to use space: *use* given data; *express* values; and *separate*, *order*, and *align* regions. In the first case, the spatial layout is given, whereas in the other four the use of space is chosen. Using the data as given is the common case with geographic data. Although there are still many nuances in design considerations, as discussed in the cartographic literature, the fundamental use of space is constrained by this choice. This approach is also the common case when dealing with scalar, vector, or tensor spatial fields, where data is sampled at many points in the field, as in volume graphics and flow visualization. Of course, the existence of spatial data does not dictate its use as the fundamental use of space in a visual encoding; a designer may still choose to derive additional data and use space differently, as discussed below. I will not discuss this case further in this talk, where I focus on choosing the use of space. Although sometimes networks are drawn using given geographic data for node positions, the common case in graph drawing is on making choices about the use of space.

The case of expressing values spatially closely follows the discussion of marks and channels in the previous section: a quantitative attribute is encoded using the spatial position of a mark. Scatterplots are the quintessential example of expressing values in this way. In contrast, the other three uses of space pertain to establishing *regions*. Separating space into distinct regions, where each region shows something different, has major implications for how we perceive the structure of the dataset. Spatial proximity strongly implies grouping at a perceptual level, and so items in different regions are perceived as being in different categories. The regions themselves can be ordered with respect to each other, for example in a data-driven way according to an ordered attribute. Finally, regions can also be aligned to a shared baseline. Lengths and positions can be compared with higher precision between aligned regions than with unaligned regions, again for fundamental perceptual reasons. A 1D alignment is a list, while aligning in 2D yields a matrix, and in 3D a volumetric grid. In any of these cases, recursive subdivision is possible to accommodate hierarchical attribute structure.

The most extreme form of separation between regions is to divide the display into multiple separate views. There are three major approaches of combining views: showing multiple views side by side, superimposing multiple views on top of each other, and having a single view that changes over time. When superimposing multiple views as layers, they must all have a shared spatial layout. A single changing view is the common case for interactive navigation. Using multiple views side by side is a particularly powerful method because of the principle that "eyes beat memory" [11]. It is easy to compare by moving one's eyes between side by side views, where the views act as external cognitive supports. It is harder to compare a visible item to the memory of what one saw before, because of the limits of internal working memory.

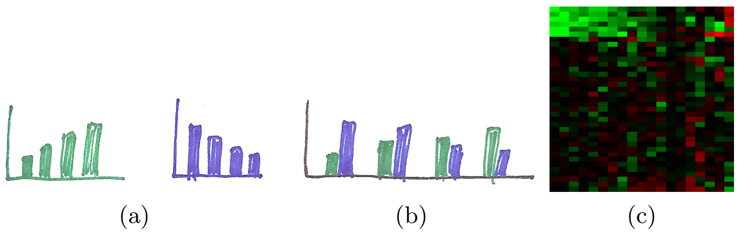

Fig. 2. Three ways to show a multidimensional table. a) Separate bar charts. b) Single interleaved bar chart. c) Heatmap. Figure credit: `http://commons.wikimedia.org/wiki/File:Heatmap.png`

Another seemingly simple example from statistical graphics is nevertheless a good example of the more complex uses of space: a multidimensional table of data with three attributes, one quantitative and two categorical. One categorical attribute is the type of export and there are two possible values: wine or cheese. The other categorical attribute is the name of the city, and there are four possible values. The quantitative attribute is the value in euro of the city's exports for a type over a year. We might encode this table as two separate bar charts, one for wine and one for cheese, with simple line marks in each, as in Figure 2a. An alternative is one interleaved bar chart where each item of data is depicted with two marks side by side,as in Figure 2b. We now have the vocabulary to analyze this choice in more detail in terms of channels and attributes: with separate charts, we have first separated into two large regions based on one categorical attribute, export type, then separated each of those into four smaller regions based on the other categorical attribute, city name. These subregions are aligned, and within each a single line mark expresses a value. The subregions are also ordered, for example either alphabetically by the name of the city, or in a data-driven way by the value of quantitative attribute, the exports. With interleaved charts, there is only one level of separation into regions: there are four regions, one for each item. Each region shows information about two attributes, as two side-by-side marks. The value of description at this level of detail is that we can reason about what kinds of information can be easily perceived by the viewer based on the partition into regions. In the first case, with two separate simple charts, the partition into one region for each type of export allows the easy perception of trends for that type. In the second case, having the two marks side by side allows the easy comparison of the export mix for a particular city.

We can now consider a quite different visual encoding of a heatmap display, which shows exactly the same data abstraction: a table with one quantitative attribute indexed by two categorical attributes. In a heatmap, regions are separated and aligned into a 2D matrix, and within each cell of the matrix an area mark is used in conjunction with the color channel to encode a quantitative attribute. The scale of the data is different: there are dozens or hundreds of values for each categorical attributes. In Figure 2c, these are genes and experimental conditions, and the quantitative attribute is the expression level of a gene in a specific condition.

I now turn from table to network datasets. A matrix view of a network is essentially the same as a heatmap, in terms of both data abstraction choice and use of space. The transformation at the data abstraction level is to transform the original network into a table, where a list of the nodes in the graph is used as both of the categorical attributes, and the weighted edge between a pair of nodes is the quantitative attribute. Thus, a cell in the matrix shows the presence or absence of an edge.

In contrast, the most common case in graph drawing is to use link marks to explicitly show links. As the name suggests, all node-link diagrams are an instance of using link connection marks. These diagrams best support tasks that pertain to the topological structure of networks [7], for example path tracing. In tree drawing, containment is also used; for example, all treemap variants are an instance of using link containment marks. These diagrams typically use the size channel to encode attribute values, either just for leaf nodes or recursively for interior nodes as well, and thus support tasks that pertain to understanding those attribute values.

In addition to the five choices for the use of spatial channels, it is useful to consider the orientation of the spatial axes within the layout. The two most common cases in graph drawing are *rectilinear* and *radial*. In 2D, the rectilinear choice yields Cartesian coordinates and the radial choice is polar coordinates. A third choice is to orient all of the axes *parallel* to each other. The limitations of these choices have been investigated in the visualization literature. Rectilinear layouts have the obvious scalability issue that the number of axes is highly constrained. A 2D rectilinear layout allows very high-precision perception of information encoded with spatial position. While 3D rectilinear layouts are possible, there are many perceptual problems in using three dimensions of space to encode nonspatial data [11]. Four or more rectilinear axes cannot be directly encoded. There has also been empirical work to investigate the strengths and weaknesses of radial layouts, in light of the known limitation that we perceive angles with less accuracy than lengths [5].

The work of McGuffin and Robert is a good example of analyzing many different tree drawing methods according to the efficiency of their use of space [8]. The *information density* of a diagram is as a measure of the amount of information encoded vs. the amount of unused space; there is a tradeoff between encoding as much information as possible, and the potential for visual clutter or other legibility problems. The examples that they analyze can be considered in terms of the methods that I have covered: whether connection or containment link marks are used, whether the layout is rectilinear or radial, how the spatial position channels are used to encode information. The information encoded in these diagrams is the link relationships, the depth in the tree of a node, and the order of siblings. Other analysis considerations are whether some information is encoded redundantly, for example through both spatial position and explicit link marks, and whether any arbitrary information is expressed through the use of spatial ordering. For example, in some trees, sibling order is not specifically defined, yet there is a visible spatial order.

Force-directed placement is a widely adopted approach for visually encoding network datasets. The visual encoding is in some sense straightforward because it is a node-link diagram: point marks represent nodes, and connection marks represent links as lines. However, considering the meaning of spatial position is somewhat tricky, because no meaning is directly encoded; instead it is left free to minimize crossings. Thus, the semantics of proximity are mixed: sometimes it is meaningful, for example when a cluster of nodes placed near to each other truly reflects strong interconnections of links between them. Sometimes it is an arbitrary artifact of the layout algorithm, and two nodes that happen to be nearby in one layout may be quite far in another. There is also an interesting tension between proximity cues and edge length: long edges are more visually salient than short ones that connect nodes close to each other.

A great deal of work has been devoted to developing better algorithms for force-directed placement through multilevel methods; the sfdp algorithm is one example [6]. The data abstraction is more complex, namely a compound graph: in addition to the original network dataset, the additional derived data of a cluster hierarchy atop the original network is computed. Although this hierarchy is used within the algorithm, it is not shown explicitly within the drawing. Thus, the fundamental use of space is the same as with simpler versions of force-directed placement; this multilevel approaches is an example of a better algorithm for the same visual encoding technique.

The GrouseFlocks system for the interactive analysis of compound graphs [1] has some instructive similarities and differences from sfdp. The data abstraction is the same, a compound graph. However, the visual encoding is different, as shown in Figure 3c. In addition to connection marks for the network links

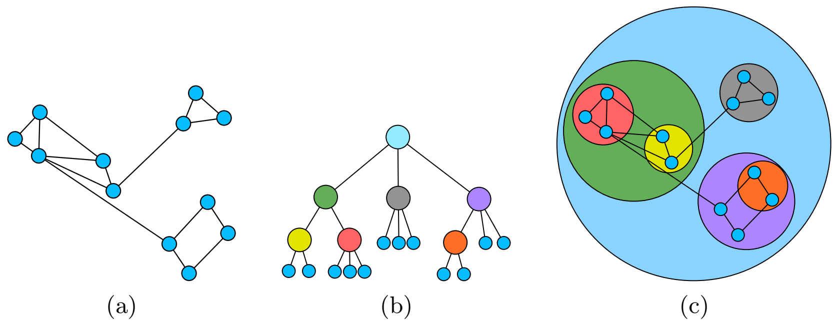

Fig. 3. Compound graph representation in GrouseFlocks [1]. a) Original network. b) Cluster hierarchy. c) Visual encoding in tool, fully expanded.

and point marks for the nodes, the hierarchy links are shown with containment marks. The system features dynamic interaction in order to support large datasets, where typically only an interactively-selected subset of the full compound graph is shown. The user can expand and contract individual metanodes in the hierarchy, which also shows the associated nodes from the base network.

6 Further Analysis Examples

In the talk, I will analyze three more systems within this framework: Cerebral [3, 4], Constellation [12, 15], and Noack's LinLog energy model [16]. All three are examples of design motivated by explicit prior analysis of the use of space.

The Cerebral system [3, 4] features both multiple side by side views, and superimposed layers within each view. The network layout is designed to mimic the semantics of hand-drawn diagrams of biological networks. I will analyze its visual encoding and design choices pertaining to the use of space in detail.

The Constellation system [12, 15] features a complex multi-level linguistic network that is laid out with spatial position reflecting specific attributes, where edge crossings are resolved using perceptual layers rather than through algorithmically reducing the number of instances.

Noack's LinLog energy model is designed to reveal clusters in data, by requiring that edges between clusters are longer than those within [16]. Noack specifically indicates that his approach uses the same minimization algorithms as previous work, and frames the energy model in terms of its visual results. I thus consider it a contribution at the visual encoding technique level, even though that exact vocabulary out of the visualization community does not appear in the paper. I note that it was published at a previous Graph Drawing conference, in contrast to the many other examples in this talk that come out of the visualization literature. I encourage more papers like this that can act as bridges between the communities!

7 Conclusions

I have discussed a general framework for systematically analyzing visualization techniques in terms of the methods of using space, and applied it to graph drawing examples in particular. It relies on breaking down visual encodings into marks that are geometric primitives representing either nodes or links, and channels that control their appearance to encode attributes. In this talk I focus on the channels related to the use of space. The simple case is using spatial position to express a quantitative attribute value, but space can also be separated into regions that partition according to a categorical attribute to indicate groups via proximity. These regions can also be ordered and aligned. This framework is a mix of ideas of that are widespread in the visualization literature and those that are new; for a more detailed discussion of the previous work, see the existing chapter [13] and the forthcoming full book [14]. These sources also describe principles in detail, and also the more complete analysis framework from which the subset discussed here was drawn. The full framework includes the nonspatial channels in addition to the spatial ones and a much more detailed discussion of methods for combining multiple views. It covers interactive techniques in addition to visual encoding techniques, particularly in terms of methods for the reduction of the amount of data shown.

This kind of analysis can guide the development of new techniques, or be used to characterize existing ones. While sometimes it is easy to map from a specific algorithm to the visual encoding technique that it supports, for example when the mapping is explicitly discussed in a paper or in work that it directly cites, sometimes this mapping is difficult to reverse-engineer. Algorithm descriptions may not facilitate analysis of the resulting visual encoding, either for the use of space or for other channels. In these cases, the line between technique and algorithm can be blurry: does a new algorithm support an existing technique, or does it result in a new one? Carrying out more such characterization may facilitate the transfer of algorithms from the graph drawing community to the visualization community. It is also important to characterize mappings between the other levels of visualization design [9], but this important question is beyond the scope of this talk.

Of course, characterization according to this sort of framework is only one of many possible ways to analyze graph drawing and visualization approaches. Benchmarks and complexity analysis are a different way to compare approaches, as are user studies in the form of controlled experiments or more qualitative investigation of how people use visualization systems [10].

References

[1] Archambault, D., Munzner, T., Auber, D.: GrouseFlocks: Steerable exploration of graph hierarchy space. IEEE Trans. on Visualization and Computer Graphics 14(4), 900–913 (2008)
[2] Auber, D.: Using Strahler numbers for real time visual exploration of huge graphs. In: Intl. Conf. Computer Vision and Graphics, pp. 56–69 (2002)

[3] Barsky, A., Gardy, J.L., Hancock, R.E., Munzner, T.: Cerebral: A Cytoscape plugin for layout of and interaction with biological networks using subcellular localization annotation. Bioinformatics 23(8), 1040–1042 (2007)

[4] Barsky, A., Munzner, T., Gardy, J., Kincaid, R.: Cerebral: Visualizing multiple experimental conditions on a graph with biological context. IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis 2008) 14(6), 1253–1260 (2008)

[5] Diehl, S., Beck, F., Burch, M.: Uncovering strengths and weaknesses of radial visualizations - an empirical approach. IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis) 16(6), 935–942 (2010)

[6] Hu, Y.: Efficient and high quality force-directed graph drawing. The Mathematica Journal 10, 37–71 (2005)

[7] Lee, B., Plaisant, C., Parr, C.S., Fekete, J.D., Henry, N.: Task taxonomy for graph visualization. In: Proc. AVI Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization (BELIV). ACM Press (2006)

[8] McGuffin, M.J., Robert, J.M.: Quantifying the space-efficiency of 2D graphical representations of trees. Information Visualization 9(2), 115–140 (2010)

[9] Meyer, M., Sedlmair, M., Munzner, T.: The four-level nested model revisited: Blocks and guidelines. In: Proc. Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization (BELIV) (2012)

[10] Munzner, T.: A nested model for visualization design and validation. IEEE Trans. Visualization and Computer Graphics (TVCG) 15(6), 921–928 (2009)

[11] Munzner, T.: Visualization Principles. VizBi 2011 Keynote (2011), `http://www.cs.ubc.ca/~tmm/talks.html#vizbi11`

[12] Munzner, T.: Constellation: Linguistic semantic networks. In: Interactive Visualization of Large Graphs and Networks (PhD thesis), ch. 5, pp. 105–122. Stanford University Department of Computer Science (2000)

[13] Munzner, T.: Visualization. In: Shirley, P., Marschner, S. (eds.) Fundamentals of Graphics, ch. 27, 3rd edn., vol. ch. 27, pp. 675–707. AK Peters (2009b)

[14] Munzner, T.: Visualization Analysis and Design: Principles, Abstractions, and Methods. AK Peters (to appear, 2014)

[15] Munzner, T., Guimbretière, F., Robertson, G.: Constellation: A visualization tool for linguistic queries from MindNet. In: Proc. IEEE Symposium on Information Visualization (InfoVis), pp. 132–135 (1999)

[16] Noack, A.: An energy model for visual graph clustering. In: Liotta, G. (ed.) GD 2003. LNCS, vol. 2912, pp. 425–436. Springer, Heidelberg (2004)

[17] Sedlmair, M., Meyer, M., Munzner, T.: Design study methodology: Reflections from the trenches and the stacks. IEEE Trans. Visualization and Computer Graphics (TVCG) 18(12), 2431–2440 (2012)