10

Scalability Considerations for Multivariate Graph Visualization

T.J. Jankun-Kelly, Tim Dwyer, Danny Holten, Christophe Hurter, Martin Nöllenburg, Chris Weaver, and Kai Xu

Scalability in visualization is a challenge: How do we choose to show more items than can be easily rendered upon a screen or understood by a human effectively? Multivariate graph visualization adds additional wrinkles in that nodes and edges are no longer atomic entities. Rather, they are repositories for further rich information. In information seeking, the mantra attributed to Ben Shneiderman succinctly outlines a path to visual scalability: "Overview first, zoom, then details-on-demand" [69]. While this is good guidance, naively presenting the whole universe of data as an initial "overview", leads to dense, unreadable displays (Fig. 10.1). To provide insightful visualizations at large scale for multivariate graphs, we must understand what our visual, cognitive, and architectural limits are, then explore approaches to mitigate these limitations. Detailed views must offer useful affordances for navigation to other views. The goals of this chapter are to identify the challenges and the state-of-the-art in these areas.

At large scale, dense multivariate graphs devolve into *hairballs* (dense collections of nodes with heavily over plotted edges) or *snowy wastes* (highly populated matrix diagrams with visually random structure) if the entire structure is shown (Fig. 10.1). Perceptual and cognitive psychology outline what human visual and mental limitations interfere with understanding such dense views; additionally, there are hardware factors which band the amount of graph data that can be rendered in a timely matter. By understanding these limitations, outlined in Sect. 10.1, designers can utilize the strategies explored in Sect. 10.2 to show only what is needed when it is needed. Use of these strategies, and further studies on the limits of scalability, are also presented in Sect. 10.3 as a means to guide further research. We conclude with a summary of challenges in scalable, multivariate graph visualization.

A. Kerren et al. (Eds.): Multivariate Network Visualization, LNCS 8380, pp. 207–235, 2014.

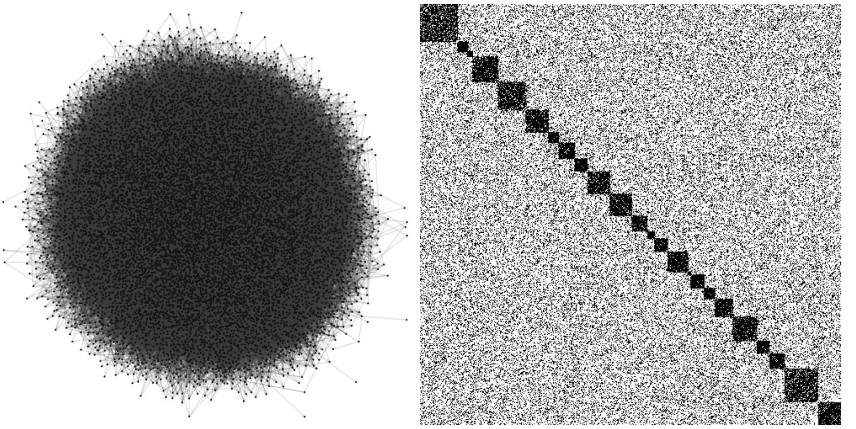


Fig. 10.1. Large (especially scale-free) graphs turn in to *hairballs* which make understanding structure difficult (left). Matrix views can help, but have limited ability to convey transitive structure at multiple levels and devolve into *snowy wastes* (right).

10.1 Limits of Visualization

When attempting to display and understand data, there are limits to what the can be humanly perceived and understood and what can be computed and displayed. To work around these limitations, they must be first understood. In this section, we examine both perceptual and computational/hardware based limitations to set the stage for the larger discussion of scalability in the next sections.

10.1.1 Limits of Visual Acuity

As a sensor, the eye has several different resolving powers or *acuities*. These acuities are measured based upon *visual angle*, the angle the viewed object(s) subtends with respect to the eye. Ophthalmologists recognize four main acuities: detection acuity (the smallest size an object can be before it cannot be seen), recognition acuity (smallest size at which an object can be identified), resolution acuity (smallest distance between two objects before they seem to merge), and localization acuity (smallest amount of visual change that can be measured between two visible objects) [87]. Perception literature in visualization has focused on the latter two, especially on point acuity (resolvability of two adjacent points), stereo acuity (the ability to resolve objects at depth), and vernier acuity (the ability to determine if two line segments are collinear) [76] (Fig. 10.2); point acuity is a form of resolution (or ordinary) acuity and stereo and vernier are localization acuities (*hyperacuities*). Resolution/point acuity are the primary acuities from the

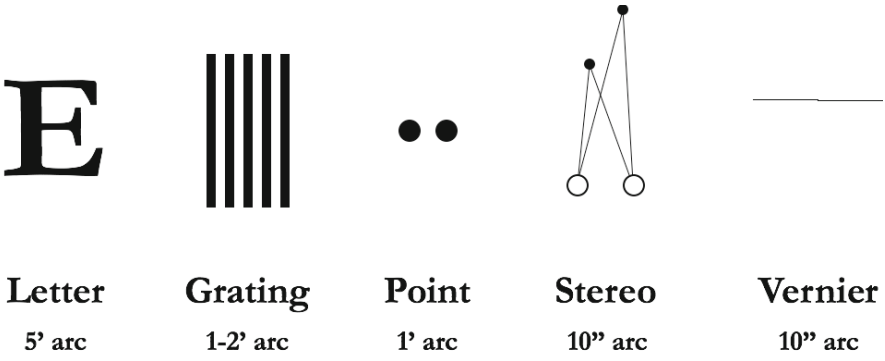


Fig. 10.2. Important acuities in visualization (after Ware [76])

standpoint of graph visualization design—it is important that two separate edges, nodes, or matrix elements be resolvable.

Assuming perfect vision, standard point acuity is one arc minute per cycle—i.e., two lines are perceived as distinct when one arc minute separates them.[1] Thus, roughly two pixels per arc minute would generate a maximally point resolvable display.[2] Though this pixel density will not be able to perfectly resolve hyperacuities, which are resolvable to 10 arc seconds, antialiasing can be used to effectively resolve hyperacuities to sub-pixel resolution [76]. At a viewing distance of 57 cm, this pixel density corresponds to 121 ppcm (pixels per centimeter); at reading distance, the density is 229 ppcm. If we relax this constraint to allow point acuity sufficient for legal driving in most countries, about half that of perfect acuity [55, 74], the pixel density would need to be one pixel per arc minute, or about 60 ppcm and 118 ppcm at viewing and reading distances respectively. For comparison, the Retina displays from Apple vary from 128 ppcm on the iPhone 5 (generally used closer to reading distance) to 87 for the Retina display laptops [88].

Given these minimum requirements for perceptual resolvability, node density (the number of nodes on the screen per area) cannot exceed roughly 1 per 2 pixels if 1 pixel representations are to be used for maximum discrimination; note, this limits our ability to indicate multivariate nodes/edges to using only the visual variables of hue and luminance. Thus, to use the MacBook Retina display as an exemplar, graphs exceeding roughly 2 million nodes would strain resolvability assuming the entire display was used and no connectivity information was displayed. If a node-like representation is used, however, this number drops to 1 million as edges must connect elements already on display. For matrix diagrams, the maximum limit is about 0.5

[1] "Perfect acuity" here is taken as an accepted average; this acuity will vary over a population.

[2] Two pixels per minute are needed to satisfy sampling theory—if we want to visually detect one pixel per arc minute, we need twice that many to satisfy the Nyquist criterion [24].

million nodes so neighboring connections are resolvable; if this is not desired, a packed representation still only represents about 1.4 million nodes as the matrix diagram's symmetric display scales quadratically with nodes. These patterns are summarized in Fig. 10.3. Exceeding these numbers means that the individual elements of the graph cannot be separated. However, staying below these numbers is not sufficient for comprehension of the graph—though elements may be perceivable, they may still exhaust cognitive resources as discussed next.

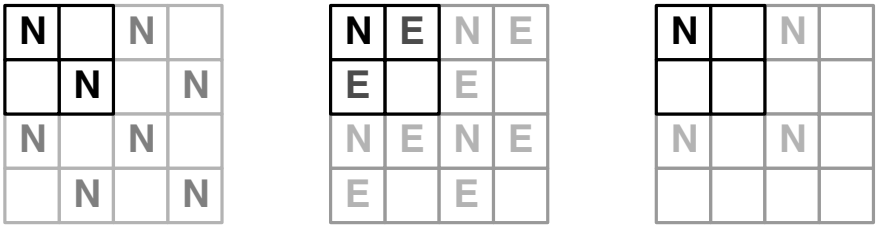


Fig. 10.3. Maximum descriminability for unconnected graphs (left), node-link diagrams (middle), and matrix diagrams (right). Each box is a pixel.

10.1.2 Cognitive Limits

Perception can be thought of grossly as a two stage process where elements are first imaged by the eye-sensor and then understood by the brain. This latter, cognitive step of perception has limitations not tied to the processing power of the eye discussed previously. Instead of focusing on raw node/element density as previously discussed, this section focuses more upon what limits of combined elements such as hue and luminance can be used, especially when used conjunctively as is often the case in multivariate visualization.

The cognitive stage of perception can be modeled as a hybrid bottom-up/top-down approach [75, 89, 90]. In the bottom-up stage, early vision separates perceived elements into feature maps (e.g., hue, orientation, length) with varying level of granularity. Elements in these maps are then compared to neighbors to measure differences. The user-driven top-down stage then searches these maps using the differences to find features of interest. As an example, if node luminance in a graph matrix display encodes weight, differences in luminance would be used as the perceptual search criteria. Thus, the levels of difference that can be encoded in the feature maps limit these lower level cognitive comparisons.

Ward et al. [75] summarize the absolute discriminability of perceptual features relevant to visualization, based upon work by Miller [54]. For item size, four to five different levels are accurately gauged; 10 levels for hue and 5 for luminance; and roughly 7 levels for line length and 8 for orientation. These

levels are not independent when combined for multi-feature encoding (a popular approach in multivariate graph visualization)—only about 12 different combinations of hue/luminance can be separated (as opposed to 40–50 if they were independent) and roughly 17 levels for combinations of hue, luminance, and size. Thus, the number of perceptual values that can be used for absolute judgements is very limited—hairballs and snowy wastes will quickly exceed these capacities.

As cognition happens at multiple levels, studies have also investigated how graph features such as paths are comprehended at latter stages in cognition. These studies provide guidance on what approaches are suitable for multivariate graph visualization. A common theme of these is the limitation of what is displayed simultaneously. A variety of studies have investigate visual search of graphs, either of specific properties such as shortest paths or for more general search. In node-link representations, path comprehension suffers when edge crossings occur [60, 61], especially when dealing with crossings over shortest paths [79]. Limiting such crossing require reducing edges or novel layouts, both approaches discussed later. Using 2.5D displays can also help graph structure comprehension when used appropriately in node-link diagrams [12, 78]; here, occlusion helps limits what is displayed. Increasing visual separation in combination with limited motion cues can also help visual search in graphs [77]. Structure can be perceived with matrix diagrams by removing edges, with more accurate reading at larger graph sizes [30].

10.1.3 Leveraging the Graphics Card (GPU)

In visualization, the human is only part of the process—computation is also required to generate the visualization. Just as there are limitations on the human, there are limitations on the computation. These can be limitations on the display technology, the graph data store, or on the means of computation on the graph. In this work, we focus on how graphics processors can enable scalable graph visualization and their limitations; for discussion on the limits and capabilities of displays and graph storage, we refer the reader to the relevant literature [2, 22].

This section describes how graphics cards can be used to address scalability issues in general and with respect to multivariate graph visualization (MGV) in particular. Sections 10.1.3 and 10.1.3 serve as an intro to and history of fixed-functionality and programmable graphics hardware as well as available programming APIs, respectively. This is followed by Sect. 10.1.3 describing typical tasks and/or application areas within graphics and (information) visualization related to MGV scalability issues. This section also gives real-world examples of GPU-based solutions designed to tackle MGV scalability issues from the perspective of rendering, interaction, and calculation.

GPU Pipeline—Fixed vs. Programmable

The rise of special-purpose graphics hardware for the accelerated monochrome and color display of 2D/3D raster and vector graphics began during the mid to late 1970s and widespread consumer adoption—especially of hardware 3D-acceleration solutions—was obtained during the late 1990s. Such hardware was originally built around "fixed functionality pipelines" (FFPs), i.e., special purpose hardware that supports a limited and fixed set of instructions (drawing commands) to display various types of graphics primitives. Typical FFPs support operations such as geometric transformation, lighting, and rasterization, all of which are necessary for displaying ("projecting") 3D graphics on 2D raster displays.

An example of early 2D/3D-accelerated FFPs are the processing pipelines on special-purpose hardware built by Silicon Graphics International (SGI) for use in their high-end graphics workstations during the early to mid 1990s. The late 1990s to early 2000s saw the widespread consumer adoption of more affordable mainstream 2D/3D-accelerated graphics FFPs (often used in game consoles) such as the Voodoo, early GeForce, and early Radeon graphics hardware by 3Dfx, NVIDIA, and ATI, respectively.

From the mid 2000s onwards, graphics hardware manufacturers as well as graphics API developers (see Section 10.1.3) gradually shifted their focus to programmable pipelines instead of FFPs. Programmable pipelines allow the graphics processing unit (GPU) to run proprietary code [58]. Such code can be used to implement new types of drawing commands and can even be used—although initially indirectly—to perform (non-graphics-related) computational tasks on a GPU, i.e., "general purpose computation on the GPU" or GPGPU [72]. The latter is useful because of the massive parallelism offered by GPUs as well as the ease with which GPUs generally handle vector and matrix operations; a direct result of the fact that 2D/3D transformations and projections within FFPs rely heavily on vector/matrix math.

GPU Programming—APIs and Pitfalls

Programming each level of the graphics card pipeline can be performed through different languages, such as NVidia's Cg, Microsoft's High-Level Shading Language (HLSL), and the OpenGL shading language (GLSL). Other specialized languages exist to do specific data processing: CUDA, OpenCL. If we exclude specific data processing languages (CUDA and OpenCL) which use specific data structures, output data must be stored in image textures. Graphics cards propose massive parallel computing but some pitfalls must be avoided in order to take advantage of this worthy power:

- Graphics card are optimized to compute data in parallel and therefore sequential algorithms cannot be paralyzed without insuring data integrity (memory protection). Reading and writing graphics memory is not possible

at the same time; this avoids memory corruption (one process reading at the same time another is updating the information). Synchronization features such as mutex or memory protection (atomic functions) much be avoided as much as possible. Specific computation technics can be applied such as MapReduce, a programming model for processing large data sets with a parallel, distributed algorithm on a cluster [35].

- Bottlenecks exist within the GPU processing, especially when transferring data between the CPU and the GPU. When this occurs, the graphics card needs to wait until every process has ended, and then start the memory transfer—a dramatically slower process. Memory transfer between the GPU and the CPU must be limited as much as possible.
- Many other pitfalls must be taken into account regarding every language, such as texture coordinates that differ between OpenGL and DirectX, debugging issues, and graphics card crashes that hinder the development process.

Multivariate Graph Visualisation (MGV) Scalability Issues

When dealing with large MGV, we identified three types of scalability limitations which are related to the InfoVis pipeline stages [6]: rendering, computation and interaction.

MGV can embed numerous items which need to be displayed. On the rendering stage, specific rendering techniques can be used to emphasize the rendering process and thus to improve data perception (an example is given in Fig. 10.4).

MGV can contain complex data structures. Layout algorithms, graph simplifications, data aggregations can be computed. Processing such information at the geometry level can be an issue when dealing with large MGV.

Finally, MGV can face scalability issues with interactive tools. Large dataset can hinder the data exploration process with system slowing down.

Instances of GPU Usages for MGV

Given the above, we identified the following GPU usages to address scalability issues with large MGV. The key to these techniques is how they overcome the limitations of the GPU mentioned previous to facilitate multivariate graph exploration—they use multi pass read-write cycles, minimize CPU-GPU memory transfer, and accommodate variation in graphical hardware:

Rendering: Graphics cards can render numerous items on the screen and thus can display large datasets. In the following examples, GPUs are used to display data and to perform image based rendering techniques. Auber developed Tulip [3], an information visualization framework dedicated to the analysis and visualization of relational data. This software uses GP-GPU techniques to render large multivariate graphs. McDonnel et al. [53] developed a framework and

an application using shaders to display multivariate data based on the dataflow model with a final image based stage. In this final step, the multivariate data of the visualization are sampled in the resolution of the current view. A more specific rendering technique is used by Holten [39] to improve edge visualization by an interesting variation on standard alpha blending, i.e. how color transparency is combined. Sheepens et al. [67] used the GPU to compute density maps and then apply shading techniques to emphasize multivariate data on the density map of moving vessels.

Computation: Graphics cards can perform fast and parallel data processing, and be used to process information at the data level. Hurter et al. [46] use the GPU for interactive exploration of multivariate relational data. Given a spatial embedding of the data, in terms of a scatter plot or graph layout, the moleview uses a semantic lens which selects a specific spatial and attribute-related data range. The lens keeps the selected data in focus unchanged and continuously deforms the data out of the selection range in order to maintain the context around the focus. Animation is also performed between the bundled and the unbundled layout of a graph. Kernel Density Edge Bundling (KDEB) [44] computes bundled layouts of general graphs. For this, KDEB first transforms a given graph drawing into a density map using kernel density estimation. Next, it applies an image sharpening technique which progressively merges local height maxima by moving the convolved graph edges into the height gradient flow. This technique is also applied on dynamic graphs [45]. Graph bundling and the computation of its density has been investigated [51], and the GPU has been used directly for graph layout as well [26].

Interaction: Interaction is an important manipulation paradigm to perform data exploration. Graphics cards can be used to provide tools to help user to interact with large datasets. ScatterDice [17] helps the user to define the appropriated displayed variables with a smooth animation when changing visual configuration; GraphDice [5] uses the same paradigms but with graph. FromDaDy [47] uses related animation with GP-GPU techniques. In order to address dataset size issue, FromDaDy loads the whole dataset within the graphics card, so that when changing visual configuration, no memory transfer is needed. This helps to improve interaction with a fast and continuous animations. Furthermore, a GP-GPU technique is implemented to support brushing and data manipulation across multiple views. One can then brush trajectories, and with a pick and drop operation he or she can spread the brushed information across views. This interaction can be repeated to extract a set of relevant data, thus formulating complex queries. Each trajectory has a unique identifier. A texture (stored in the graphics card) contains the Boolean selection value of each trajectory. When the trajectory is brushed its value is set to true. The graphics card uses parallel rendering which prevents reading and writing in the same texture in a single pass. Therefore FromDaDy used

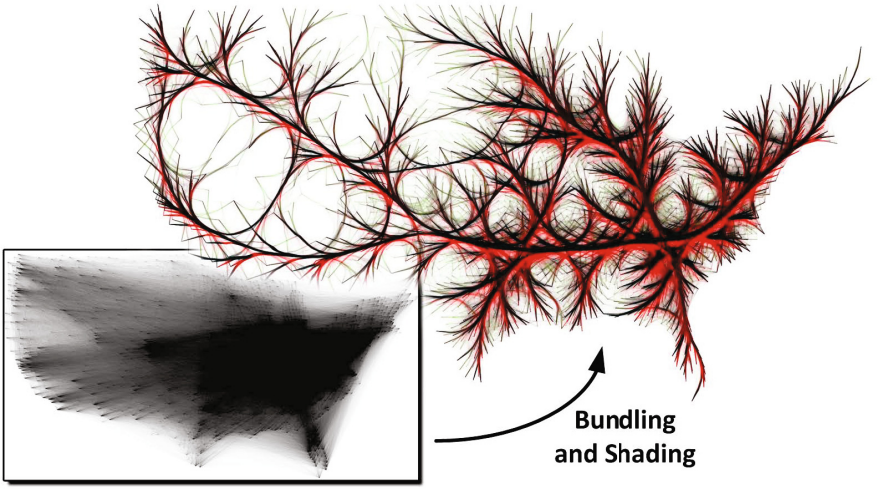


Fig. 10.4. County-to-county migration flow, (1091764 nodes, the Census 2000): people who moved between counties within 5 years. Original data only shows the outline of the USA (bottom), bundled [44] and shaded path (top) shows multiple information like East-West and north-South paths, shading shows data density.

a two-step rendering process: firstly it tests the intersection of the brushing shape and the point to be rendered to update the selected identifier texture, and, secondly, it draws all the points with their corresponding selected attribute (gray color if selected, visual configuration color otherwise).

10.2 Design Strategies for Scalable Multivariate Graph Visualization

The perceptual, cognitive, and technical factors presented in Sect. 10.1 limit the scalability of network visualization in general. In particular, we described the limitations of:

- visual acuity (10.1.1);
- human cognition (10.1.2);
- computer hardware (10.1.3); and
- computability of aesthetic and clear layout (10.1.2 and 10.1.3).

When faced with the increased amount and complexity of information that one typically encounters in multivariate networks, it is necessary to address scalability of visualization by additional means. For example, there may be rich tabular data associated with graph elements; graph elements may have myriad types; and graphs may be derived from underlying data in many different ways. In this section, we review various design strategies to support scalable interactive visualization of multivariate graphs, including very large ones. These strategies go beyond simply getting as much information onto the

screen as possible. They also aim to make good use of available display real estate by transforming and reducing that information to facilitate exploration and analysis.

Chapter 6 describes interactive operations in terms of the information visualization reference model of Card et al. [7]. In the model, three transformation steps connect a progression of four modes of data representation from raw data (at one end) to displayed visuals (at the other end) (Fig. 6.1). In keeping with the reference model, we organize multivariate graph design strategies into the three following categories of transformations of information representations.

Data transformation and reduction strategies provide alternative *network compositions* by being selective about the type of structure and amount of information to show. These strategies use combinations of aggregation, projection, and filtering techniques to convert multivariate graph data sets into other data sets, particularly into alternative multivariate graphs having topologies and attributes that can be more readily and usefully displayed.

Visual mapping strategies provide alternative *network presentations* by mapping data dimensions and values into visual elements that efficiently communicate graph structure and multivariate attributes. The definition of *efficient* here depends on the application and the type of analysis being sought. These strategies often complement data transformation and reduction strategies by choosing mappings to suit the aggregated, projected, and filtered information of specific network perspectives.

View transformation strategies provide alternative *network perspectives* by providing a feedback loop for the analyst to interact with visual elements and the space in which they are shown. Visualizing multivariate networks in any real analysis application is not a static "batch" or "pipeline" process. View transformation strategies often complement other strategies by supporting not only navigation in view space and selection of data items, but also interactive changes to the functions and parameters used in data transformation and reduction and in visual mapping.

Weaver breaks down this progression of transformations into a more detailed model that specifically targets interactive visualization of complex multivariate data as networks. The model is implemented in Improvise [82] and has been used in a variety of graph visualization applications including to meta-visualize multiple view coordination structure [83, 84] and to analyze individual differences in user categorization of starplot shapes [49] and geospatial relationships [50]. Figure 10.5 depicts the data transformation pipelines in this graph reference model, as customized for use in the Attribute Relationship Graph technique [86]. In this model, the three transformation steps in Card's model are expanded into three interdependent phases of transformation.

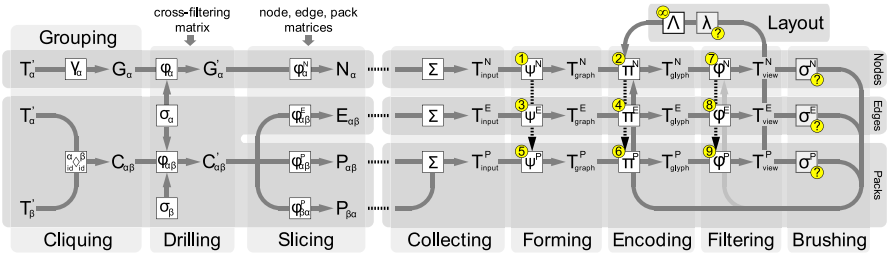


Fig. 10.5. The data transformation pipeline of Weaver [86] showing transformation of raw multivariate data into tractable graph views

Card's Data Transformation step expands into the first two phases: data projection and graph definition. The data projection phase (Fig. 10.5, left) aggregates occurrences (*Grouping*) and co-occurrences (*Cliquing*) of the data values in each dimension, then determines which occurrences and co-occurrence to show as a function of chosen dimensions (*Drilling*) and data values (*Slicing*). Graph definition consists of transformations (Fig. 10.5, center) that gather the aggregates into tables (*Collecting*) that are then mapped into a graph representation (*Forming*) consisting of primitive elements identified as nodes, edges and "packs". (Packs can be thought of as hyperedges that connect multiple nodes into semantically grouped aggregates, and are often shown as convex hulls around the connected nodes, as in Vizster [36]).

Card's Visual Mapping and View Transformation steps expand into a single graph visualization phase (Fig. 10.5, right). Parallel pipelines that take the sets of graph primitives generated in graph definition as input. The subsequent *Encoding*, *Filtering*, *Layout*, and *Brushing* transformations populate and support interaction with graph primitives in network, matrix, and other data views. Transformations are interdependent to capture the ways that one can expect graph elements to be coordinated in appearance and behavior, such as filtering of edges on whether their nodes are visible (as well as as a function of their co-occurrence data attributes), or reencoding edges when a node moves during automatic or interactive layout.

Together, the three strategies can be seen as axes in a rich design space. We use the graph reference model here as a frame of reference to discuss the work that has been done in this space. Below we explore each of the strategies in more detail, looking at how they are employed individually and in combination in various exemplar systems.

10.2.1 Data Transformation and Reduction

Visual bandwidth is finite, both on the production side (the graphics and display hardware), and on the consumption side (the perceptual and cognitive capabilities of the person trying to analyze the data); these are alluded to

in the previous section. Data transformation and reduction techniques aim to use the available visual bandwidth to support foraging, sense-making, and insight by showing only parts of the data and from particular viewpoints. A variety of data transformation and reduction strategies are routinely and usefully employed for increasingly sophisticated visually querying of network structure in data sets, in systems such as Jigsaw [70], Coordinate Graph Visualization (CGV) [73], Ploceus [52], Orion [37], and Candid [68].

Whatever the data-size limitation in a certain setting is, as soon as the graph exceeds it, we can no longer show all information in a single static view. Instead, data reduction techniques must be applied to extract a task-specific neighbourhood of the larger graph for display. This extraction can be fully automatic, or semi-automatic according to the constraints of the user. As in Chapter 6, we are interested in data reduction operations necessary for producing limited views of very large graphs. In this chapter, however—with our focus on scalability—we review recent work that deals more specifically with the problem of extracting small tractable graph views from big tabular data. We distinguish three different data transformation and reduction approaches: aggregation, projection, filtering.

Aggregation Techniques

Graph aggregation techniques transform and reduce data sets by collecting data records into buckets in terms of commonalities shared by the raw or derived attributes of those records. The underlying principle of graph aggregation is, for a given graph $G = (V, E)$, to derive an aggregate graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ with fewer vertices and edges. The goal is to compute $\widetilde{G}$ in such a way that it is a good coarse representation of G for the user's data analysis purposes. This data reduction process is also known as *granulation* [71].

In the graph reference model, graph aggregation happens in the grouping (for vertices) and cliquing (for edges) stage. These transformations perform unary and binary calculations to determine raw or derived attributes occurrences and co-occurrences, respectively. Using raw attribute values themselves is a basic approach but still highly useful for analysis; Jigsaw [70], Cross-Filtered Views [85], and Attribute Relationship Graphs [86] all support multivariate association an comparison tasks in this way.

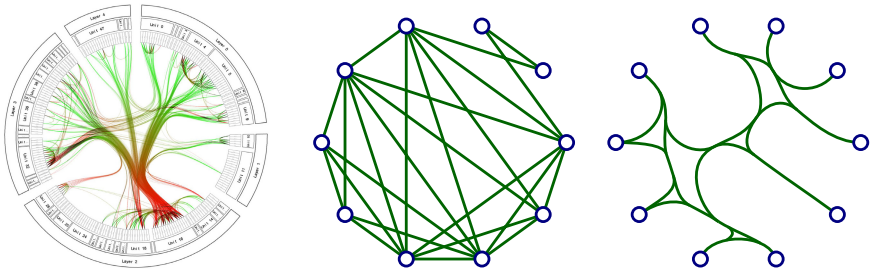
Many systems provide several types of graph aggregation that entail more complex calculations of derived graph elements (i.e., with a one-to-many mapping from graph node or link to data). PivotGraph [81] uses roll-up for multivariate graphs to group nodes into equivalence classes based on their attribute values and create weighted edges as induced by the members of the different equivalence classes. Selection by restricting certain attribute values can be used to obtain the induced subgraphs. This technique originally does not focus on large graphs and depending on the attribute types and values, the number of equivalence classes may be too high. Orion [37] supports similar attribute-based aggregation of vertices for networks that are

obtained in a previous step from relational database tables. In the Ploceus system [52], three types of aggregation-by-attribute are identified: *pivoting* is equivalent to PivotGraph's *roll-up* but specifically for categorical attributes; *binning* is used to describe grouping nodes by quantitative attributes divided into distinct ranges; *proximity grouping* is used to refer to more sophisticated clustering techniques involving distance functions of quantitative attributes.

More generally, one can perform arbitrary many-to-many calculations to generate derived data dimensions for grouping and cliquing. Clustering is a common approach, although one that requires great care to maintain responsiveness of the overall graph transformation pipeline. *Graph clustering* techniques partition the vertex set V into mutually disjoint clusters $C_1, \ldots, C_k$ with the objective that two vertices in the same vertex cluster C_i are sufficiently similar and two vertices from different clusters are sufficiently dissimilar. As we deal with multivariate graphs, there is a wide range of clustering methods that can be applied.

Graph-based clustering methods consider edges (possibly with weights) as an indicator for similarity and hence aim at finding a clustering with high intra-cluster edge density and low inter-cluster edge density. Fortunato's recent survey on community detection in graphs covers the state of the art in clustering algorithms [25]. On the other hand, attribute-based clustering methods are data mining techniques that consider each vertex as a point in a multi-dimensional space spanned by the multivariate vertex attributes. Using a (dis-)similarity measure defined in this space, clusters are derived based on this measure. Again, vertices in the same cluster should have high similarity, and vertices in different clusters should be dissimilar. Berkhin gives a recent review of the most common clustering methods in data mining [4]. Clustering of multivariate graphs ideally uses methods that combine connectivity information and attribute information in a configurable way, especially if attributes and edges are not highly correlated. Only few methods exist that take into account both types of information. Zhou et al. [94, 95] present a method to transform the attribute data of large graphs into additional graph edges and then apply a graph clustering algorithm to the augmented graph. Another combined method is DB-CSC by Günnemann et al. [31], which allows more flexible cluster shapes. Hadlak [33] describes clustering on time series behavior of time-varying attributes.

Of particular interest are hierarchical graph clustering methods [25, Chap. IV.B], where different clustering granularities can be represented between a single cluster containing everything at the top and singleton clusters at the bottom. In the graph reference model, hierarchies can be treated as multiple aggregations that are coincidentally related in terms of data type semantics. (In the graph visualization phase of the model, representation and interaction should reinforce these relationships.) Depending on the navigation strategy, different types of cuts or *frontiers* in the clustering tree can be applied. Thus it is possible to obtain rather uniform granularities for an aggregated overview graph or non-uniform granularities giving more details in a focus region and



(a) An example of hierarchical edge bundling [39].

(b) A straight-line drawing (left) and a strict confluent drawing (right) of the same graph [20].

Fig. 10.6. Radial graph layouts using edge aggregation

less details in the further context. ASK-GraphView [1] is a system that applies hierarchical clustering for visualizing large graphs.

While the above methods are mostly concerned with aggregating vertices to reduce the graph size, there are also several techniques to aggregate edges. Since a graph of $|V|$ vertices can have $O(|V|^2)$ edges, graphs with relatively few vertices can already be too edge dense to be readable. Edge bundling methods [21, 27, 39, 40, 44, 45] aim to reduce visual clutter by visually grouping together edges between similar parts of the graph thus using fewer pixels to show the original set of edges. Visualizations using edge bundling are well suited to depict global connectivity patterns with reduced visual complexity. See Fig. 10.6(a) for an example. The topological information produced in the graph specification phase of the graph reference model can be used to aggregate edges, although having two or more (potentially interdependent) stages of aggregation complicates matters substantially; such complex interdependencies between the node and edge pipelines that define a graph visualization are beyond the scope of the graph reference model.

The concept of confluent graph drawing [10, 18–20, 43, 63], where two vertices are connected if and only if there is a smooth path between them, similarly merges and splits the curves representing edges in a visualization. But unlike edge bundling methods, confluent drawings are unambiguous or *information faithful* [57] since no false adjacencies are created. Confluent drawings can be used to display certain non-planar graphs without edge crossings. Figure 10.6(b) shows a straight-line and a confluent drawing of the same graph. Confluent drawing algorithms are not yet implemented in practical systems and some related decision problems are known to be NP-complete. Edge compression through Power Graph Analysis [13, 66] (see Fig. 10.7) is another technique for aggregating edges by replacing the edges of bipartite clique sub-graphs with single edges connecting the two sets of nodes in the bipartite clique. Power Graph compression is related to confluent drawing in that it also offers an unambiguous, information faithful representation of the original dense graph however practical techniques exist for their generation [16].

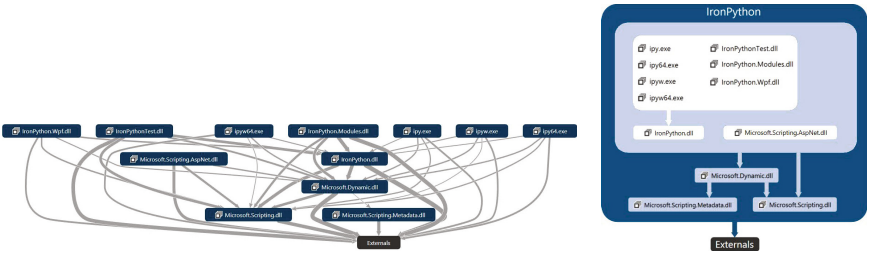


Fig. 10.7. Illustration of edge compression to simplify dense graphs. 10.7(a) the top-level component graph produced by the Visual Studio code-dependency analysis tool for the IronPython code base with 39 edges. 10.7(b) a power-graph decomposition of the same graph leaves only six aggregate links without any loss of connectivity information for an 85% compression rate.

Projection Techniques

The Ploceus system [52] is concerned with allowing users a multitude of ways to extract graph views of tabular data. A key part of their system is a *network schema* view of the rich heterogenous graphs that can easily be obtained from such data. The network schema shows a graph of the types of nodes in the network and the types of links that are allowed between pairs of such nodes. The network schema view thus provides a powerful affordance for restricting the set of nodes and edges shown in the actual network visualization. That is, the user can select a subset of the available node and edge types that will appear in the network visualization. An important concept in realizing these final network views is *projection*. That is, for the final subset view to be usefully representative of the original graph, nodes that are to be omitted from the final view must be *spliced* out of the network, rather than simply removed, potentially leaving the graph disconnected even though a transitive relationship exists.

Filtering Techniques

In contrast to aggregation techniques, *graph filtering* selects an appropriately sized subgraph of the input graph, either by stochastic sampling or by deterministic processes. Typically, an importance function measures the relevance of vertices and edges in the graph and only the most relevant objects are kept. Selections can be done by computing additional vertex and edge attributes, e.g., centrality measures, that indicate how important these features are in the graph [48]. Van Ham and Perer [34] use a degree-of-interest function to determine the relevant subgraph for one or more focus points in the graph. This function evaluates both the graph topology and the multivariate graph attributes based on the selected focal vertex. Subsequently, a

maximal interest subgraph of specified size is extracted. By interacting with the visualization, users can expand additional parts of the graph that they are interested in.

10.2.2 Visual Mapping

Once a sub-graph is chosen for actual visualization by application of the data reduction techniques above, there are further scalability considerations in the Visual Mapping stage. By "Visual Mapping" we mean the visual representations of data sets as views and/or visual encodings of data items in views. Multiple views can help with orienting the user of the visualization system when the displayed visuals represent only a very small fraction of the total data space. For example, the display may be configured to show an abstract over-view of a large portion of the full graph while a detailed view shows a much more restricted neighborhood but with many more attributes shown on each of the visible nodes and edges [14]. In general, this mapping happens in the encoding and filtering stages of the pipeline. For multiple views, multiple encoding operations coexist, e.g., Ploceus and Attribute Relationship Graphs both have a central graph view that feeds off of all three encoding operations (for nodes, edges and packs) and peripheral views (fed by node or edge encoding operations).

As already discussed in this chapter, various paradigms exist for visual mappings for graphs, the two most widely known being node-link diagrams and matrix views. The limits of scalability for each of these were discussed in Sect. 10.1 while more exotic representations are discussed in Chap. 7. The appeal of node-link diagrams is that it is fairly natural for most people to illustrate related concepts by connecting labels with lines, and—at least while the diagram is simple enough to be unambiguous—for readers of such diagrams to follow transitive paths. By contrast, matrices offer unambiguous representations of very dense graphs (i.e., graphs with a high proportion of edges to nodes). Henry et al. [38] have elegantly demonstrated that hybrid visual mapping may offer the best of both representations. In their NodeTrix system, they use matrices to display dense parts of a large graph, while these matrices are themselves treated as nodes situated within a larger node-link diagram.

Encodings include input on graph element position from the layout feedback loop of the pipeline. Layout operates on the filtered graph subset and changes the position encoding of elements. Complex encoding operations support rich visual mappings, for example: edge centric schemes such as those suggested by Riche et al. [64] control edge curvature based on edge attributes.

In general, there is a trade-off between scalability of layout techniques and the quality of the resultant drawing. For node-link diagrams, algorithms exist that can obtain layouts that may be useful to show the gross structure of an overall graph for thousands (even hundreds of thousands of nodes) in reasonable time [26, 32, 41]. However, for small diagrams—especially when the

nodes are not just points but also need to display multiple attributes—there are additional and computationally expensive considerations for layout, for example: avoiding overlaps between node boundaries [15, 28] and minimizing edge-edge and node-edge crossings [59].

10.2.3 View Transformation

The use of data reduction to limit the view to only a small sub-graph—perhaps a sub-graph that is specifically chosen for a particular line of enquiry—necessitates flexible navigation affordances, to allow analysts to easily refocus on different aspects or parts of the graph. We call this type of navigation *view transformation*. For example, Huang et al. [42] developed an early system for exploring an *infinite* graph by browsing just a small neighbourhood at a time. A simple animated spring algorithm enabled incremental layout as nodes are added to or removed from the neighbourhood, and gave the graph smooth transitions. Fisher coined the term "ego-centric views" [23] to describe views of the graph from one particular node's point of view, or from a small neighborhood.

In the pipeline of Fig. 10.5, this is the chief concern of the brushing stage, where encoding can depend on brushing to highlight (un)selected graph elements, while filtering can depend on brushing to elide (un)selected graph elements. The brushing stage also covers view-specific interaction capabilities, such as panning and zooming to navigate a graph coordinate space. In the full pipeline model (and implementation in the Improvise system [82]), operations at *all* stages can depend on navigation and selection visualization parameters controlled throughout a coordinated multiple view visualization.

Overview+detail can be thought of as branching late in the pipeline with different levels of filtering controlling the portions of graph visible in each view, and more encoding to show increased detail (e.g., attributes) of remaining elements. Robert's *Multiform Visualization* [65] idea boils down to multiple pipelines with varying encoding and filtering.

Focus+context techniques also show the most detail around only a small neighborhood, but they endeavor to show this neighborhood in the context of the larger graph. Focus+context graph techniques can be thought of combinations of visual mapping transformations with multiple views that are nested. A compelling and scalable example of this design concept is the *Topological Fisheye* technique of Gansner et al. [29] and another that takes advantage of graphics hardware is proposed by Zinsmaier et al. [96]. In the topological fisheye system a layout is computed for the entire graph, then a combination of spatial and structural clustering techniques are used to show an abridged view of the graph with only gross detail visible. The authors describe navigation in which the user is able to zoom in to show full detail in a small focal region with the abridged contextual structure still visible.

10.3 Studies on Scalability in Graph Visualization

This section provides a summary of the evaluations related to multivariate graph scalability. Some of them are part of work that has been discussed so far. Use Cases are the most popular form of study, but in many works, these cases were designed to demonstrate the proposed technique rather than being a formal evaluation. This section covers some of the case studies, but the focus is on the formal studies, both qualitative (such as interviews) and quantitative (such as controlled experiments).

10.3.1 Data Transformation and Reduction

Wattenberg [81] described the "pilot usage" of "PivotGraph" (please see Sect. 10.2.1 for details) in his paper. These are essentially observations followed by semi-structured interviews after participants have been using the tool for a considerable period. The results are from five analysts who looked for new patterns in their own data using PivotGraph. They are very familiar with the data, which have been analyzed with other tools.

Three multivariate graphs were used in the study: the first one is a transition matrix consisting of 521 states (nodes) and 2,671 transition probabilities (weighted edges). Besides the edge weighting attribute, each node (state) had four associated categorical attributes. The second dataset is the social network among a community of 146 people within a large company. Each person (node) in the community was classified on five dimensions. The last dataset is similar to the second one: it is the communication patterns among employees of a company, with each employee classified according to five different dimensions. The graphs used in the study is not small. For instance the transition matrix graph had 521 nodes and 2,671 edges. However, due to the aggregation technique deployed in PivotGraph, the number of nodes shown in the examples were less than 100 nodes. In that sense, visual complexity is well under control.

The paper provided detailed description of how PivotGraph was used to analyze these datasets, especially what the new findings were and how they were discovered. This provides support for the claim that PivotGraph can help identify new patterns in multivariate graph data. All the participants are very positive about their experience of using the PivotGraph, and they especially liked the feature that allows quick visual comparison between different pair of dimensions (attributes). All the participants wanted to continue using the tool, together with what they were using already. This shows that PivotGraph can be an useful addition to the multivariate graph analysis tool collection.

The Orion system paper [37] includes three use cases: online medical forum discussions, academic collaborations, and software developments. The first use case involves 3 million discussion posts from `MedHelp.org`. The analyst was able to construct network based on edge weight to answer relevant

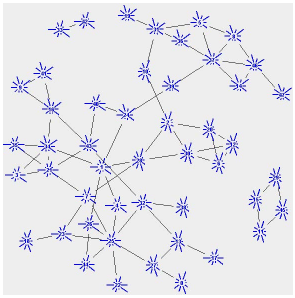
questions. The resulting visualization led to discovery of errors in the dataset and interesting co-occurrence of forum participants on different medical topics. The second use case used the publication information from the ACM Digital Library to visualize the career development of academics. The last use case is based on the Github data and the visualization showed the difference between the followers to the cities where open-source development are most active.

10.3.2 Visual Mapping

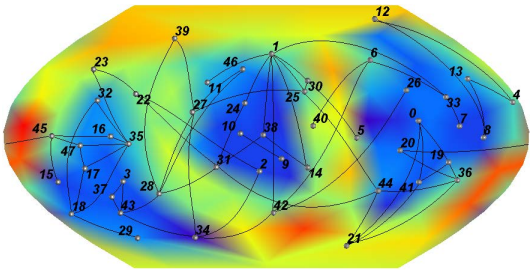
Wu and Takatsuka conducted an user study [91] to evaluate the effectiveness of their multivariate networks visualization method that uses Self-Organizing Map (SOM) to improve its layout. Their method tries to find optimal node distance based on not only graph distance but also graph attribute similarity. The evaluation consisted two parts: two use cases and a controlled experiment. The first use case is a student friendship network, with node attribute being the result of two courses. There are 43 nodes and 55 directed edges in total. The results showed that it was possible to achieve good balance between node attribute clustering (measured by "data distortion") and graph drawing aesthetics (measured by "edge crossings") by adjusting their weighs in the SOM function.

The second use case is based on the Krackhardt's high-tech manager advice network [80]. Again, this is a relative small social network with 21 nodes (the managers) but dense connections (190 edges). Each manager has four attributes: Age, Tenure, Position Level, and Department. The results again showed it is possible to achieve a good balance between the attribute clustering and layout aesthetics. The user study compared their method (Fig. 10.8(b)) with a glphy-base one (Fig. 10.8(a)), in which a star glyph is used to show node attributes. It involved 33 participants performing tasks on 7 synthesis multivariate networks. These networks had between 30-50 nodes, 40-70 edges, and 4 or 10 attributes. The tasks included comparing the set of neighbors of two given nodes in terms of their attribute similarity and comparing relationships within the same set of entities. The results showed that the participants spent more time using the glyph-based visualization, which also had lower accuracy.

A study by Cunningham et al. [9] evaluated their method of visualizing multivariate network using 2.5D surfaces, each of which represents a node attribute. They compared their method, GraphScape [92] (Fig. 10.9(a)), to the approach of using node size to show the attribute value (Fig. 10.9(b)). In the first experiment, the participants were asked to select the 20% nodes with the largest attribute value from the visualization of graphs with up to 100 nodes (Fig. 10.9(b)). The results showed that there was no significant difference in accuracy between the two methods, but it took longer to complete the task with the GraphScape. In the second experiment, the participants were asked to determine the average value of a variable for a cluster of nodes

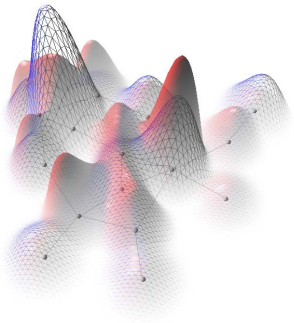


(a) Star Glyph: each node is a star glyph to show its attributes.

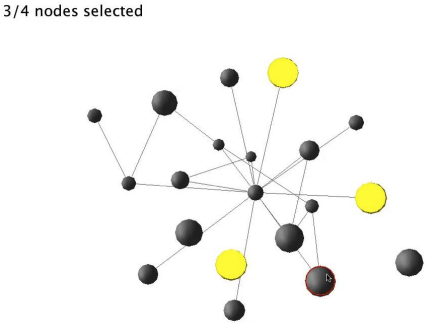


(b) Self-Organizing Map-based Hybird Layout: nodes are placed not only to reduce edge crossings but also to show their attribute similarity.

Fig. 10.8. The two multivariate network visualizations used in the user study by Wu and Takatsuka [91] in their paper on hybrid layout method



(a) GraphScape: node attributes are shown as 2.5D surfaces. Two attributes are shown as red and blue surface in this example.



(b) The other visualization used in the study: using node size to show attribute value. The task is to select the top 20% nodes with the largest value.

Fig. 10.9. The visualizations used in the evaluation of the GraphScape [92] method

using both visualizations, with similar-sized graphs. The accuracy of GraphScape was found to be significantly greater than that of using node size. However, participants answered significantly faster with node-size visualization, comparing to GraphScape.

There is an increasing usage of curved edges in graph visualization techniques designed to address scalability issues, such as the edge boundling methods discussed in Sect. 10.2.1. There were two user studies [62, 93] on the impact on readability when using curved edges in graph visualization. Edge curvature can be used to encode edge attribute, and it is commonly used in the edge bundling and confluent drawing methods discussed earlier.

The first study consisted of two experiments. The first experiment examine the impact of three different curvature levels on graph readability, with the straight edges (zero curvature) included as the baseline. Participants completed path-finding tasks in a controlled experiment setup, and the graphs used have 20, 50, or 100 nodes. The results showed that using either straight edges or slightly curved edges are more accurate and faster than using heavily curved edges. There was no significant difference in accuracy between the straight edge and the slightly curved edge, but the former is significantly faster. The graph size had a significant impact on speed (each size increase incurred a significant time penalty), but less so on the accuracy. The second experiment included force-directed Lombardi layout [8], which uses circular edges to maximize angular resolution (the minimal angle between edges adjacent to a node). Four tasks were tested in the experiment and the largest graphs have 200 nodes. There was no significant accuracy difference but both straight edge and Lombardi layout were faster than the slightly curved edge. The study by Purchase et al. compared the two variations of the Lombardi layout with straight-edge graphs produced by force-directed method. The size of the graphs used was smaller (20 or 40 nodes) but each size had two edge density level. The three tasks were similar to the second experiment in the study by Xu et al. discussed above. The results were quite different from the previous study: straight edges were found to be faster and more accurate than the two variations of the Lombardi layout. The user preference was also different: Lombardi layout was the choice for aesthetics in this study whereas straight edge was the preferred option in the study by Xu et al.

10.3.3 Navigation and Interaction

Dörk et al. designed a visualization method, PivotPaths [11] (Fig. 10.10), to allow browsing of large data collections through their multiple facets and encourages exploration and serendipitous discoveries. While the method is not designed with multivariate network in mind, it provides a novel way to interactively visualize the relationships between data records through the similarity among their attributes. Because of the design goal, Dörk et al. decided to use a longitudinal study together with observation and semi-structured interviews. The data set used is a collection of academic publications with 160,000 articles, 180,000 authors, and 20,000 keywords. The study was conducted in a research institute with more than 200 recorded user sessions, which were followed by interviews with four participants. The data from the recorded sessions and comments from the interviews confirmed that the PivotPaths provided an integrated view of the three facets in the data (publications, authors, and keywords) and the relationships among them. The participants found the "pivoting" animation is easy to follow and it encouraged them to explore more about the dataset. However, there was someone confusion about pivoting and filtering: some participants expected filtering when they "pivoted" from one facet to another.

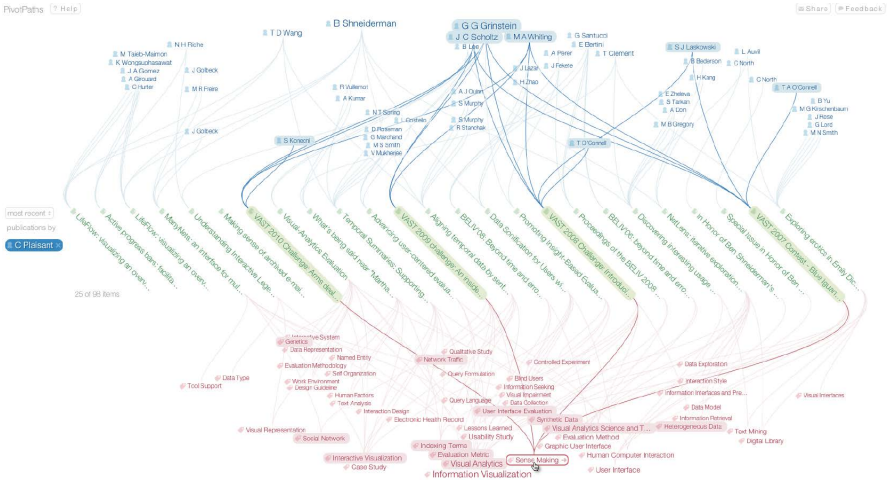
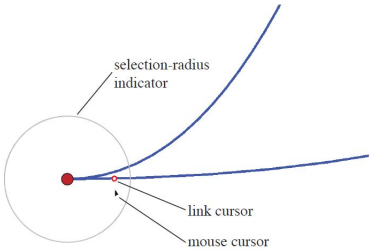
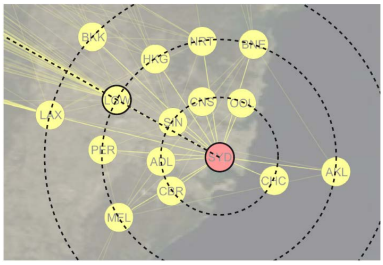


Fig. 10.10. PivotPaths showing the links between the three facets of research publications: author (top), paper (middle), and topic (bottom)

The work by Tomer et al. [56] introduced two navigation techniques, Link Sliding (Fig. 10.11(a)) and Bring & Go (Fig. 10.11(b)), for large networks. Both adopted the focus+context approach and were evaluated in a controlled experiment. The experiment used two randomly generated scale-free graphs, one sparse (1,000 nodes; 1,485 edges), and one dense (1,000 nodes; 2,488 edges). There were 12 participants and the tasks include identifying all nodes connected to a given node, following a link, and returning to a previously visited link (revisit task). Besides the two new techniques, "Pan and Zoom" and "Bird's Eye View" were also included. The Bring & Go technique is significantly faster than Pan-and-Zoom and Bird's-Eye-View in all tasks, but Link Sliding was not significantly faster than Bird's-Eye-View in two out of the three tasks. There was no significant difference in accuracy for the "following" and "revisit" task because there were very few errors. The participants unanimously agreed that Bring & Go was quick, and made the tasks easy. They also found it the least tiring, and most pleasant to use. Link Sliding and Bird's Eye View both received mixed comments regarding the ease and speed at accomplishing tasks. Pan & Zoom was generally rated as slow and difficult to use for the given tasks. Because the two networks used in the experiment shared the same number of nodes, it is not possible to observe how the techniques scale with graph node number. However, edge density did show negative performance impact for some of the tasks. The two techniques were not designed for multivariate networks and may require extra work to provide such support.



(a) Link Sliding allows the sliding along a (long) edge when the cursor is within the *selection radius*.



(b) Bring & Go makes all the neighbors, some of which normally would be outside the frame, visible within the display.

Fig. 10.11. The Link Sliding and Bring & Go method designed for navigating large graphs

10.4 Challenges and Future Directions

We have attempted to review the state-of-the-art from research and industry in addressing the problem of scalability for multivariate graph visualization; limitations from the hardware and cognitive side were also overviewed. Hopefully, the principles and design guidelines discussed will be useful to implementers of new systems. It should be noted that the systems reviewed here tend to be either research prototypes or visualization platforms specifically designed for a particular type of graph or application. The "holy-grail" of a visualization system that can be easily applied to any type or amount of multivariate graph data remains very much an open challenge. However, the so-called "big-data" problem is ever growing with the steady march of Moores' law and the growth of the internet. Similarly, there is a growing popularity of a network view of data, evidenced by the rise of technologies such as social networking, so-called "graph search", and a move away from tabular data paradigms for storage, such as graph databases. For these reasons we think that more and more researchers and practitioners will begin to explore the use of visualization for very large multivariate graph data and we expect to see rapid developments in this area in the future.

References

1. Abello, J., van Ham, F., Krishnan, N.: ASK-GraphView: A large scale graph visualization system. IEEE Transactions on Visualization and Computer Graphics 12(5), 669–676 (2006)
2. Angles, R., Gutiérrez, C.: Survey of graph database models. ACM Comput. Surv. 40(1) (2008)
3. Auber, D.: Tulip: A huge graph visualisation framework(2003); Mutzel, P., Junger, M. (eds.), `http://hal.archives-ouvertes.fr/hal-00307626`

4. Berkhin, P.: A survey of clustering data mining techniques. In: Kogan, J., Nicholas, C., Teboulle, M. (eds.) Grouping Multidimensional Data, pp. 25–71. Springer, Heidelberg (2006), http://dx.doi.org/10.1007/3-540-28349-8_2
5. Bezerianos, A., Chevalier, F., Dragicevic, P., Elmqvist, N., Fekete, J.D.: Graphdice: A system for exploring multivariate social networks. Comput. Graph. Forum 29(3), 863–872 (2010)
6. Card, S.K., Mackinlay, J.D., Shneiderman, B.: Readings in information visualization: Using vision to think. Morgan Kaufmann Publishers Inc. (1999)
7. Card, S.K., Mackinlay, J.D., Shneiderman, B. (eds.): Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann (January 1999)
8. Chernobelskiy, R., Cunningham, K.I., Goodrich, M.T., Kobourov, S.G., Trott, L.: Force-directed lombardi-style graph drawing. In: Speckmann, B. (ed.) GD 2011. LNCS, vol. 7034, pp. 320–331. Springer, Heidelberg (2011)
9. Cunningham, A., Xu, K., Thomas, B.H.: Seeing more than the graph – evaluation of multivariate graph visualization methods. In: Proceedings of the Workshop on Interactive Data Exploration and Knowledge Discovery (Part of International Working Conference on Advanced Visual Interfaces 2010), Rome, Italy, pp. 429–429 (2010)
10. Dickerson, M., Eppstein, D., Goodrich, M., Meng, J.: Confluent drawings: Visualizing non-planar diagrams in a planar way. Journal of Graph Algorithms and Applications 9(1), 31–52 (2005)
11. Dörk, M., Riche, N., Ramos, G., Dumais, S.: PivotPaths: strolling through faceted information spaces. IEEE Transactions on Visualization and Computer Graphics 18(12), 2709–2718 (2012)
12. Dwyer, T.: Two-and-a-half-dimensional Visualisation of Relational Networks. Ph.D. thesis, School of Information Technologies, Faculty of Science, University of Sydney (2005)
13. Dwyer, T., Henry Riche, N., Marriott, K., Mears, C.: Edge compression techniques for visualization of dense directed graphs. IEEE Transactions on Visualization and Computer Graphics 19(12), 2596–2605 (2013)
14. Dwyer, T., Marriott, K., Schreiber, F., Stuckey, P., Woodward, M., Wybrow, M.: Exploration of networks using overview+ detail with constraint-based cooperative layout. IEEE Transactions on Visualization and Computer Graphics 14(6), 1293–1300 (2008)
15. Dwyer, T., Marriott, K., Stuckey, P.J.: Fast node overlap removal. In: Healy, P., Nikolov, N.S. (eds.) GD 2005. LNCS, vol. 3843, pp. 153–164. Springer, Heidelberg (2006)
16. Dwyer, T., Mears, C., Morgan, K., Niven, T., Marriott, K., Wallace, M.: Improved optimal and approximate power graph compression for clearer visualisation of dense graphs. In: PacificVis 2014, pp. 105–112. IEEE (2014)
17. Elmqvist, N., Dragicevic, P., Fekete, J.D.: Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. IEEE Trans. Vis. Comput. Graph. 14(6), 1148–1539 (2008)
18. Eppstein, D., Goodrich, M.T., Meng, J.Y.: Delta-confluent drawings. In: Healy, P., Nikolov, N.S. (eds.) GD 2005. LNCS, vol. 3843, pp. 165–176. Springer, Heidelberg (2006)
19. Eppstein, D., Goodrich, M.T., Meng, J.Y.: Confluent layered drawings. Algorithmica 47(4), 439–452 (2007)
20. Eppstein, D., Holten, D., Löffler, M., Nöllenburg, M., Speckmann, B., Verbeek, K.: Strict confluent drawing. In: Wismath, S., Wolff, A. (eds.) GD 2013. LNCS, vol. 8242, pp. 352–363. Springer, Heidelberg (2013)

21. Ersoy, O., Hurter, C., Paulovich, F., Cantareiro, G., Telea, A.: Skeleton-based edge bundling for graph visualization. IEEE Transactions on Visualization and Computer Graphics 17(12), 2364–2373 (2011)
22. Fikkert, W., D'Ambros, M., Bierz, T., Jankun-Kelly, T.J.: Interacting with visualizations. In: Kerren, A., Ebert, A., Meyer, J. (eds.) Human-Centered Visualization Environments 2006. LNCS, vol. 4417, pp. 77–162. Springer, Heidelberg (2007)
23. Fisher, D.: Using egocentric networks to understand communication. IEEE Internet Computing 9(5), 20–28 (2005)
24. Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F.: Computer Graphics: Principles and Practice in C, 2nd edn. Addison-Wesley (1996)
25. Fortunato, S.: Community detection in graphs. Physics Reports 486(3-5), 75–174 (2010), http://www.sciencedirect.com/science/article/pii/S0370157309002841
26. Frishman, Y., Tal, A.: Multi-level graph layout on the gpu. IEEE Transactions on Visualization and Computer Graphics 13(6), 1310–1319 (2007)
27. Gansner, E., Hu, Y., North, S., Scheidegger, C.: Multilevel agglomerative edge bundling for visualizing large graphs. In: Proc. PacificVis, pp. 187–194 (2011)
28. Gansner, E.R., Hu, Y.: Efficient node overlap removal using a proximity stress model. In: Tollis, I.G., Patrignani, M. (eds.) GD 2008. LNCS, vol. 5417, pp. 206–217. Springer, Heidelberg (2009)
29. Gansner, E.R., Koren, Y., North, S.C.: Topological fisheye views for visualizing large graphs. IEEE Transactions on Visualization and Computer Graphics 11(4), 457–468 (2005)
30. Ghoniem, M., Fekete, J.D., Castagliola, P.: On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. Information Visualization 4(2), 114–135 (2005)
31. Günnemann, S., Boden, B., Seidl, T.: DB-CSC: A density-based approach for subspace clustering in graphs with feature vectors. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part I. LNCS (LNAI), vol. 6911, pp. 565–580. Springer, Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-23780-5_46
32. Hachul, S., Jünger, M.: An experimental comparison of fast algorithms for drawing general large graphs. In: Healy, P., Nikolov, N.S. (eds.) GD 2005. LNCS, vol. 3843, pp. 235–250. Springer, Heidelberg (2006)
33. Hadlak, S., Schumann, H., Cap, C.H., Wollenberg, T.: Supporting the visual analysis of dynamic networks by clustering associated temporal attributes. IEEE Transactions on Visualization and Computer Graphics 19(12), 2267–2276 (2013)
34. van Ham, F., Perer, A.: 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)
35. He, B., Fang, W., Luo, Q., Govindaraju, N.K., Wang, T.: Mars: a mapreduce framework on graphics processors. In: Proceedings of the 17th international conference on Parallel architectures and compilation techniques, PACT 2008, pp. 260–269. ACM Press, New York (2008), http://doi.acm.org/10.1145/1454115.1454152
36. Heer, J., Boyd, D.: Vizster: Visualizing online social networks. In: Proceedings of the IEEE Symposium on Information Visualization (InfoVis), pp. 33–40. IEEE, Minneapolis (October 2005)

37. Heer, J., Perer, A.: Orion: A system for modeling, transformation and visualization of multidimensional heterogeneous networks. In: 2011 IEEE Conference on Visual Analytics Science and Technology (VAST), pp. 51–60. IEEE (2011)
38. Henry, N., Fekete, J.D., McGuffin, M.J.: NodeTrix: A hybrid visualization of social networks. IEEE Transactions on Visualization and Computer Graphics 13(6), 1302–1309 (2007)
39. Holten, D.: Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. IEEE TVCG 12(5), 741–748 (2006)
40. Holten, D., van Wijk, J.J.: Force-directed edge bundling for graph visualization. Comp. Graph. Forum 28(3), 670–677 (2009)
41. Hu, Y.: Efficient, high-quality force-directed graph drawing. Mathematica Journal 10(1), 37–71 (2005)
42. Huang, M.L., Eades, P., Wang, J.: On-line animated visualization of huge graphs using a modified spring algorithm. Journal of Visual Languages & Computing 9(6), 623–645 (1998)
43. Hui, P., Pelsmajer, M.J., Schaefer, M., Stefankovic, D.: Train tracks and confluent drawings. Algorithmica 47(4), 465–479 (2007)
44. Hurter, C., Ersoy, O., Telea, A.: Graph bundling by kernel density estimation. Comp. Graph. Forum 31(3 pt. 1), 865–874 (2012), http://dx.doi.org/10.1111/j.1467-8659.2012.03079.x
45. Hurter, C., Ersoy, O., Telea, A.: Smooth bundling of large streaming and sequence graphs. In: Proceedings of the PacificVis 2013 (2013)
46. Hurter, C., Telea, A., Ersoy, O.: Moleview: An attribute and structure-based semantic lens for large element-based plots. IEEE Transactions on Visualization and Computer Graphics 17(12), 2600–2609 (2011), http://dx.doi.org/10.1109/TVCG.2011.223
47. Hurter, C., Tissoires, B., Conversy, S.: Fromdady: Spreading aircraft trajectories across views to support iterative queries. IEEE Transactions on Visualization and Computer Graphics 15(6), 1017–1024 (2009), http://dx.doi.org/10.1109/TVCG.2009.145
48. Jia, Y., Hoberock, J., Garland, M., Hart, J.: On the visualization of social and other scale-free networks. IEEE Transactions on Visualization and Computer Graphics 14(6), 1285–1292 (2008)
49. Klippel, A., Hardisty, F., Li, R., Weaver, C.: Colour enhanced star plot glyphs – can salient shape characteristics be overcome? Cartographica 44(3), 217–231 (2009)
50. Klippel, A., Weaver, C., Robinson, A.C.: Analyzing cognitive conceptualizations using interactive visual environments. Cartography and Geographic Information Science 38(1), 52–68 (2011)
51. Lambert, A., Bourqui, R., Auber, D.: Winding roads: Routing edges into bundles. Comp. Graph. Forum 29(3), 432–439 (2010)
52. Liu, Z., Navathe, S.B., Stasko, J.T.: Network-based visual analysis of tabular data. In: Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST 2011), pp. 41–50. IEEE (2011)
53. McDonnel, B., Elmqvist, N.: Towards utilizing gpus in information visualization: A model and implementation of image-space operations. IEEE Transactions on Visualization and Computer Graphics 15(6), 1105–1112 (2009), http://dx.doi.org/10.1109/TVCG.2009.191
54. Miller, G.A.: The magical number seven, plus or minus two: Some limits on our capacity for processing information. The Psychological Review 63(2), 81–97 (1956)

55. Millodot, M.: Dictionary of Optometry and Visual Science. Butterworth-Heinemann (1997)
56. Moscovich, T., Chevalier, F., Henry, N., Pietriga, E., Fekete, J.D.: Topology-aware navigation in large networks. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2319–2328. ACM (2009)
57. Nguyen, Q., Eades, P., Hong, S.-H.: On the faithfulness of graph visualizations. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 566–568. Springer, Heidelberg (2013)
58. Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A., Purcell, T.J.: A survey of general-purpose computation on graphics hardware. Computer Graphics Forum 26(1), 80–113 (2007), http://www.blackwell-synergy.com/doi/pdf/10.1111/j.1467-8659.2007.01012.x
59. Pupyrev, S., Nachmanson, L., Bereg, S., Holroyd, A.E.: Edge routing with ordered bundles. In: Speckmann, B., van Kreveld, M. (eds.) GD 2011. LNCS, vol. 7034, pp. 136–147. Springer, Heidelberg (2011)
60. Purchase, H.C.: Which aesthetic has the greatest effect on human understanding? In: DiBattista, G. (ed.) GD 1997. LNCS, vol. 1353, pp. 248–261. Springer, Heidelberg (1997)
61. Purchase, H.C., Carrington, D., Allder, J.-A.: Experimenting with aesthetics-based graph layout. In: Anderson, M., Cheng, P.C.H., Haarslev, V. (eds.) Diagrams 2000. LNCS (LNAI), vol. 1889, pp. 498–501. Springer, Heidelberg (2000)
62. Purchase, H.C., Hamer, J., Nöllenburg, M., Kobourov, S.G.: On the usability of lombardi graph drawings. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 451–462. Springer, Heidelberg (2013)
63. Quercini, G., Ancona, M.: Confluent drawing algorithms using rectangular dualization. In: Brandes, U., Cornelsen, S. (eds.) GD 2010. LNCS, vol. 6502, pp. 341–352. Springer, Heidelberg (2011)
64. 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, pp. 506–513. ACM (2012)
65. Roberts, J.C.: Multiple-View and Multiform Visualization. In: Erbacher, R., Pang, A., Wittenbrink, C., Roberts, J. (eds.) Proceedings of SPIE Visual Data Exploration and Analysis VII, vol. 3960, pp. 176–185 (January 2000)
66. Royer, L., Reimann, M., Andreopoulos, B., Schroeder, M.: Unraveling protein networks with power graph analysis. PLoS computational biology 4(7), e1000108 (2008)
67. Scheepens, R., Willems, N., van de Wetering, H., Andrienko, G., Andrienko, N., van Wijk, J.J.: Composite density maps for multivariate trajectories. IEEE Transactions on Visualization and Computer Graphics 17(12), 2518–2527 (2011), http://dx.doi.org/10.1109/TVCG.2011.181
68. Shadoan, R., Weaver, C.: Visual analysis of higher-order conjunctive relationships in multidimensional data using a hypergraph query system. IEEE Transactions on Visualization and Computer Graphics 19(12), 2070–2079 (2013)
69. Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. In: Proceedings of the IEEE Symposium on Visual Languages, pp. 336–343. IEEE (1996)
70. Stasko, J., Görg, C., Liu, Z.: Jigsaw: Supporting investigative analysis through interactive visualization. Information Visualization 7(2), 118–132 (2008)
71. Stell, A.J.: Granulation for graphs. In: Freksa, C., Mark, D.M. (eds.) COSIT 1999. LNCS, vol. 1661, pp. 417–432. Springer, Heidelberg (1999)

72. Thompson, C.J., Hahn, S., Oskin, M.: Using modern graphics architectures for general-purpose computing: a framework and analysis. In: Proceedings of the 35th Annual ACM/IEEE International Symposium on Microarchitecture, MICRO, vol. 35, pp. 306–317. IEEE Computer Society Press, Los Alamitos (2002), http://dl.acm.org/citation.cfm?id=774861.774894
73. Tominski, C., Abello, J., Schumann, H.: CGV–an interactive graph visualization system. Computers & Graphics 33(6), 660–678 (2009)
74. Vogel, D., Balakrishnan, R.: Distant freehand pointing and clicking on very large, high resolution displays. In: Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology (UIST 2005), pp. 33–42. ACM Press, New York (2005)
75. Ward, M.O., Grinstein, G.G., Keim, D.A.: Interactive Data Visualization-Foundations, Techniques, and Applications. A K Peters (2010)
76. Ware, C: Information Visualization: Perception for Design, 2nd edn. Morgan Kaufmann (2004)
77. Ware, C., Bobrow, R.: Supporting visual queries on medium-sized node-link diagrams. Information Visualization 4(1), 49–58 (2005)
78. Ware, C., Mitchell, P.: Visualizing graphs in three dimensions. ACM Trans. Appl. Percept. 5(1), 2:1–2:15 (2008)
79. Ware, C., Purchase, H.C., Colpoys, L., McGill, M.: Cognitive measurements of graph aesthetics. Information Visualization 1(2), 103–110 (2002)
80. Wasserman, S., Faust, K.: Social network analysis: methods and applications. Cambridge University Press, Cambridge (1994)
81. Wattenberg, M.: Visual exploration of multivariate graphs. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 811–819. ACM (2006)
82. Weaver, C.: Building highly-coordinated visualizations in Improvise. In: Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2004), pp. 159–166. IEEE Computer Society, Austin (October 2004)
83. Weaver, C.: Visualizing coordination in situ. In: Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2005), pp. 165–172. IEEE Computer Society, Minneapolis (October 2005)
84. Weaver, C.: Metavisual exploration and analysis of DEVise coordination in Improvise. In: Proceedings of the International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV), pp. 79–90. IEEE Computer Society, London (July 2006)
85. Weaver, C.: Cross-filtered views for multidimensional visual analysis. IEEE Transactions on Visualization and Computer Graphics 16(2), 192–204 (2010)
86. Weaver, C.: Multidimensional data dissection using attribute relationship graphs. In: Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST), pp. 75–82. IEEE, Salt Lake City (October 2010)
87. Westheimer, G.: Visual acuity. In: Kaufman, P.L., Alm, A. (eds.) Adler's Physiology of the Eye: Clinical Applications, 10th edn., ch. 17, pp. 453–469. Elsevier (1987)
88. Wikipedia: List of display by pixel density: Apple, http://en.wikipedia.org/wiki/List_of_displays_by_pixel_density#Apple (last accessed November, 2013)
89. Wolfe, J.M.: Guided search 2.0: A revised model of visual search. Psychonomonic Bulletin & Review 1(2), 202–238 (1994)
90. Wolfe, J.M., Cave, K.R., Franzel, S.L.: Guided search: An alternative to the feature integration model for visual search. Journal of Experimental Psychology 15(3), 419–433 (1989)

91. Wu, Y., Takatsuka, M.: Visualizing multivariate networks: A hybrid approach. In: Proceedings of the IEEE Pacific Visualization Symposium (PacificVis 2008), pp. 223–230 (2008)
92. Xu, K., Cunningham, A., Hong, S.H., Thomas, B.H.: GraphScape: integrated multivariate network visualization. In: Proceedings of the 6th International Asia-Pacific Symposium on Visualization, Sydney, Australia, Febraury 2007, pp. 33–40 (2007)
93. Xu, K., Rooney, C., Passmore, P., Ham, D.H., Nguyen, P.: A user study on curved edges in graph visualization. IEEE Transactions on Visualization and Computer Graphics 18(12), 2449–2456 (2012)
94. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. Proc. VLDB Endow. 2(1), 718–729 (2009), http://dl.acm.org/citation.cfm?id=1687627.1687709
95. Zhou, Y., Cheng, H., Yu, J.: Clustering large attributed graphs: An efficient incremental approach. In: 2010 IEEE 10th International Conference on Data Mining (ICDM), pp. 689–698 (2010)
96. Zinsmaier, M., Brandes, U., Deussen, O., Strobelt, H.: Interactive level-of-detail rendering of large graphs. IEEE Transactions on Visualization and Computer Graphics 18(12), 2486–2495 (2012)