Image-Based Graph Visualization: Advances and Challenges

Alexandru Telea

Bernoulli Institute, University of Groningen, The Netherlands
`a.c.telea@rug.nl`

Abstract. Visualizing large, multiply-attributed, and time-dependent graphs is one of the grand challenges of information visualization. In recent years, image-based techniques have emerged as a strong competitor in the arena of solutions for this task. While many papers on this topic have been published, the precise advantages and limitations of such techniques, and also how they relate to similar techniques in the more traditional fields of scientific visualization (scivis) and image processing, have not been sufficiently outlined. In this paper, we aim to provide such an overview and comparison. We highlight the main advantages of image-based graph visualization and propose a simple taxonomy for such techniques. Next, we highlight the differences between graph and scivis/image datasets that lead to limitations of current image-based graph visualization techniques. Finally, we consider these limitations to propose a number of future work directions for extending the effectiveness and range of image-based graph visualization.

Keywords: Large graph visualization · Image-based information visualization · Multiscale visualization.

1 Introduction

Relational data, also called networks or graphs, is a central and ubiquitous element of many types of data collections generated by multiple application domains such as traffic analysis and planning, social media, business intelligence, biology, software engineering, and the internet. Since the first moments when such data was collected, visualization has been a key tool for its exploration and analysis, leading to the emergence and development of the research domains of *graph drawing* and *graph visualization* [15, 23]. Last-decade developments in processing power, data-acquisition tools, and techniques, have led to what is today globally called *big data* – collections of tens of millions of samples having hundreds of measurement values (attributes), all which can evolve over thousands of time steps. A particular case hereof, big-data graphs, pose fundamental problems for visual exploration.

On the other hand, several solutions, techniques, and tools have been developed for the scalable visual exploration of other types of big data collections, such

as 2D images, 3D scalar or vector field volumes, or more generally multidimensional fields, in the domains of *scientific visualization* and *imaging sciences* [52]. Recent developments have tried to approach the two traditionally separately evolving fields of graph visualization and scientific visualization, thereby aiming at leveraging the (visual) scalability of the latter methods to address big graph related challenges from the former. [20]. This has led to interesting parallels and links between concepts, methods, and applications between the two fields, and the development of hybrid visualization methods that inherit strenghts from both graph visualization and scientific visualization. However, large graph visualization still has many unsolved challenges [23].

In this paper (and related talk) we aim to provide an overview of the research at the crossroads of large graph visualization and scientific visualization. We start highlighting the main challenges in large graph visualization (Sec. 2). Next, we outline the high-level directions proposed by current research towards addressing these (Sec. 3). We focus next on one type of technique that aims to solve these challenges by adapting methods from scientific visualization and imaging to the particularities of graph visualization – image-based graph visualization (Sec. 4). Based on the structure of graph data outlined in Sec. 2, we discuss here various types of image-based methods for graph visualization and highlight parallels to simplification methods for multivariate field and image data. In the light of these methods, we next highlight open challenges for image-based graph visualization (Sec. 5) and attempt to clarify some of the more subtle points related to this new emerging visualization field which, we believe, have not been sufficiently discussed in current literature. Section 6 concludes the paper outlining promising directions for future research in image-based graph visualization.

2   Problem Definition

2.1   Preliminaries

To better outline the large graph visualization challenges, we first introduce some notations. Let $G = (V, E \subset V \times V)$ be a graph with vertices, or nodes, $V = \{\mathbf{v}_i\}$ and edges $E = \{\mathbf{e}_i\}$. Both nodes and edges typically have one or multiple attributes (also called features, dimensions, or variables). We denote by v_i^j, $1 \leq j \leq N_V$, the individual attributes of node $\mathbf{v}_i$, and by e_i^j, $1 \leq j \leq N_E$, the individual attributes of edge $\mathbf{e}_i$, respectively. As a shorthand, let $\mathbf{v}^j$ denote all values of the jth attribute of all nodes V; let $\mathbf{e}^j$ denote all values of the jth attribute of all edges E; let $\mathbf{V} = (\mathbf{v}^1, \ldots, \mathbf{v}^{N_V})$ denote all values of all node attributes; and let $\mathbf{E} = (\mathbf{e}^1, \ldots, \mathbf{e}^{N_E})$ denote all values of all edge attributes, respectively. Attributes can be of all types, *e.g.*, quantitative (values in $\mathbb{R}$), integral (values in $\mathbb{N}$), ordinal, categorical, text, hyperlinks, but also more complex data types such as images or video. In this sense, the ordered collections $\mathbf{V}$ and $\mathbf{E}$ are very similar to so-called multidimensional datasets as well known in information visualization [34, 20, 52]. That is, every node $\mathbf{v}_i$ or edge $\mathbf{e}_i$ can be seen as a sample, or observation, of a respectively N_V and N_E dimensional

dataset. Finally, as graphs can evolve over time, all their ingredients (sets V, E, $\mathbf{V}$, and $\mathbf{E}$) can be seen as functions of (continuous or discrete) time [2].
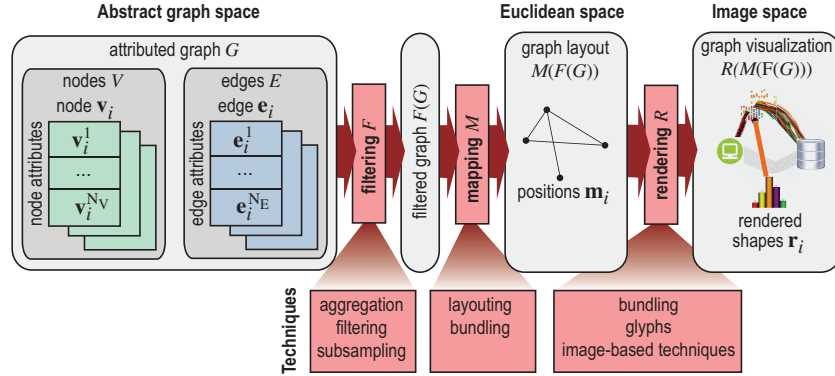

Fig. 1. Graph visualization pipeline.

With these notations, visualizing a graph can be defined in terms of the traditional data visualization pipeline [52] in terms of *filtering*, *mapping* and *rendering* operations (see also Fig. 1). Filtering F reads the input graph G and produces another graph $F(G)$ which is (more) suitable for subsequent visualization, *e.g.*, by removing nodes, edges, and/or attributes that are not of interest, or aggregating such elements into fewer and/or semantically richer ones. Mapping M is a function that takes as input $F(G)$ and outputs a set of shapes $M(F(G)) = \{\mathbf{m}_i\}$ embedded in $\mathbb{R}^2$ or, less frequently, $\mathbb{R}^3$. Typically, nodes are mapped to individual points, and edges are mapped to straight lines or, less commonly, curves. Other layout methods, such as adjacency matrices [1], exist but are less intuitive, less common, and thus not discussed here. Most often, M takes into account only the graph topology (V, E), and computes only positions $\mathbf{m}_i$ for nodes. This is the case of so-called *graph layout* techniques [55, 51]. Rendering R takes as input the layout $M(F(G))$ and creates actual visible shapes $R(M(F(G))) = \{\mathbf{r}_i\}$, where each $\mathbf{r}_i$ is placed at the corresponding layout positions $\mathbf{m}_i$. Visual variables [57] of $\mathbf{r}_i$ such as size, color, texture, transparency, orientation, texture, and annotation are used to encode the attributes $\mathbf{v}_i$ and $\mathbf{e}_i$ of the respective node or edge. Interactive exploration techniques such as zooming, panning, brushing, and lensing can be subsumed to the rendering operator R as they are essentially customized ways to perform rendering; hence, we do not discuss them separately.

2.2   Scalability Challenge

With the above notations, we can decompose the challenge of visualizing big-data graphs into the following three elements:

Layout: A good layout should arguably allow end users to detect structures of interest present in G by examining the rendering $R(M(G))$. These include, but

are not limited to, finding groups of strongly-connected nodes; finding specific connection patterns; assessing the overall topology of G; and finding (and following) paths between specific parts of G, at a low level [24]; and identifying, comparing, and summarizing the information present in G, at a high level [4]. However, even for moderately-sized graphs ($|V|$ or $|E|$ exceeding a few thousands), most existing layout methods cannot usually produce layouts that can consistently support these tasks [23]. Suboptimal layouts of large graphs, also called 'hairballs', are all to frequent a problem in graph visualization [38, 47]. The problem is caused by the fact that there does not exist a 'natural' mapping between the abstract space of graphs and the Euclidean 2D or 3D rendering space. Interestingly, the problem is very similar to that of mapping high-dimensional scatterplots (sampled datasets in $\mathbb{R}^n$) to 2D or 3D by so-called dimensionality reduction (DR) methods [50, 29].

Dimensionality: An effective graph visualization should allow users to answer questions on all elements of interest of the original graph. Apart from the topology (V, E) which should be captured by the layout $M(G)$, this includes the node and edge attributes $\mathbf{V}$ and $\mathbf{E}$. The problem is that, when N_V and N_E are large, nodes and edges essentially become points in high-dimensional spaces. Since, as explained, each node and/or edge is typically mapped to a separate location $\mathbf{m}_i$, the challenge is how to depict a high-dimensional data sample, consisting of potentially different attribute types, to the space at or around $\mathbf{m}_i$. A similar problem exists in scientific visualization when using glyphs to depict high-dimensional fields [44, 3]: The higher-dimensional our data points are, the more space one needs to show all dimensions, so the fewer such points (in our case, nodes and/or edges) can one show on a given screen size. At one extreme, we can display (tens of) thousands of nodes on a typical computer screen if we only show 2 or 3 attributes per node (encoded *e.g.* in hue, luminance, and size); at the other extreme, we can display tens of attributes per node, like in UML diagrams, but for only a few tens up to hundreds of nodes [5]. The problem is well known also in multidimensional information visualization.

Clutter and Overdraw: Finally, a scalable graph visualization should accommodate (very) large graphs consisting of millions of nodes and/or edges. Even if we abstract from the aforementioned layout and dimensionality challenges, a fundamental difficulty here resides in the fact that a node-link visualization cannot exceed a given *density*: If nodes and/or edges are drawn too close to each other, they will form a compact *cluttered* mass where they cannot be distinguished from each other. Additionally, an edge (in the node-link visual model) is drawn as a line (or curve) so in the limit it needs to use at least a few (tens of) pixels of screen space to be visible as such (if the edge is too show, we cannot *e.g.* see its direction); in Tufte's terms, there is an upper bound to the data-ink ratio [57] when drawing a graph edge. Moreover, when attributes must be rendered atop of the edge, the amount of surrounding whitespace needs to be increased [17]. This leads in turn to inherent *overdraw, i.e.* edges that partially occlude each other, even for moderately-sized graphs of thousands of nodes. A detailed overview of clutter reduction techniques in information visualization is

given by Ellis and Dix [8]. In large graph visualization, clutter and overdraw are hard to jointly optimize for: Spatial distortion, *e.g.* via edge bundling (discussed next in Sec. 4.2), creates more white space, thus reduces clutter, but increases overdraw; space-filling techniques are of limited effect since, as noted, edges must be surrounded by white space to be visible as such; apart from these, reducing clutter and overdraw is not fully possible in the rendering phase only, as this phase works within the constraints of the layout fed to it by the M operator.
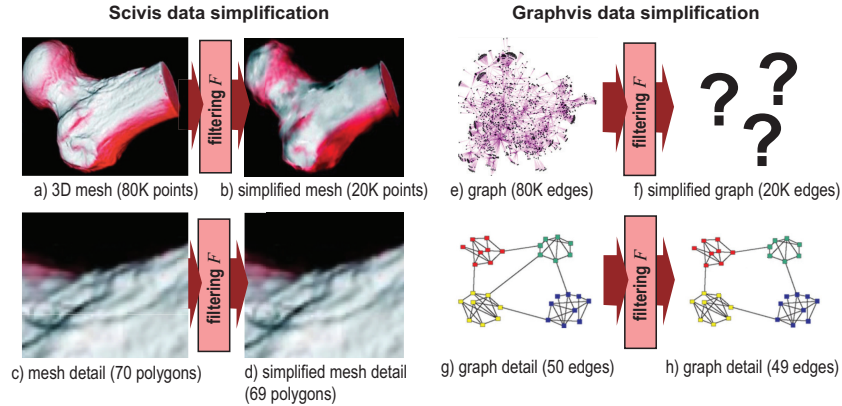
Scivis data simplification   **Graphvis data simplification**

a) 3D mesh (80K points)   b) simplified mesh (20K points)   e) graph (80K edges)   f) simplified graph (20K edges)

c) mesh detail (70 polygons)   d) simplified mesh detail (69 polygons)   g) graph detail (50 edges)   h) graph detail (49 edges)

Fig. 2. Data-space simplification of scivis data (a) *vs* graph data (b).

3 Simplification: Ways Towards a Solution

For a given screen resolution for the target image R, how can we approach large graph visualization? Given the scalability challenges outlined in Sec. 2.2, two types of approaches exist, as follows.

Data-space Simplification: First, we can simplify the graph G in the filtering stage F in the visualization pipeline (Fig. 1). This reduces the number of nodes ($|V|$), edges ($|E|$), and/or attributes (N_V, N_E) to be next passed to the mapping operator M. Following the clutter reduction taxonomy of Ellis and Dix [8], this includes subsampling, filtering, and clustering (aggregation) [45], all applicable to V, E, and $(\mathbf{V}, \mathbf{E})$ respectively. While effective in tackling clutter, overdraw, and dimensionality issues, such approaches have two limitations. First, they require *a priori* knowledge on which data items (samples or dimensions) can be filtered or clustered together. Secondly, performing such operations on graphs can easily affect the *semantics* of the underlying data.

At this point, it is instructive to compare graph visualization (graphvis) with image and field visualization as done in classical scientific visualization (scivis). Consider a multidimensional dataset $D : \mathbb{R}^m \to \mathbb{R}^n$; for each point of the Euclidean m-dimensional domain, n quantitative values are measured. Scivis provides many methods for visualizing such datasets, *e.g.* for 2D and 3D vector fields

$(m \in \{2, 3\}, n \in \{2, 3\})$ or 2D and 3D scalar fields $(m \in \{2, 3\}, n = 1)$ [52]. Many techniques exist in scivis (and, by extension, in imaging and signal processing) for simplifying large fields – we mention here just a few, *e.g.*, perceptually-based image downscaling [39], feature extraction from vector fields [43], multiscale representations of scalar and vector fields [14, 12], mesh simplification [28], and image segmentation [40]. Many such techniques have a *multiscale* nature: Given a dataset D and a simplification level $\tau \in \mathbb{R}^+$, they produce a filtered (simplified) version $F(D)$ of D which is (roughly) τ times smaller than D. This allows users to continuously vary the level-of-detail parameter τ until obtaining a visualization that matches their goals, as well as fits the available screen space with limited clutter. Figure 2(left) illustrates this: From a 3D surface-mesh dataset (a), we can easily extract a four times smaller dataset (b) using *e.g.* mesh decimation [46], which captures very well the overall structure of the depicted bone shape. Consider now a graph of similar size, whose nodes are functions in a software system [54] and edges function calls respectively (e). What should be the *equivalent* simplification of this graph to a size four times smaller? (f) This is far from evident. The scivis-graphvis difference manifests itself even on the tiniest scale: Take a detail (zoom-in) of the mesh dataset (c) from which we decimate a *single* polygon (data point). The result (d) is visually identical. Consider now the analogous zoom-in on a small portion of our call graph (g) from which we remove a *single* edge. The result (h) may be visually similar to the input (g), but can have a completely different *semantics* – just imagine that the removed function call edge is vital to the understanding of the operation of the underlying software.

All in all, most scivis data-space simplification methods succeed in keeping the overall semantics of their data. In contrast, even tiny changes to graph data can massively affect the underlying semantics. More formally put, scivis data-space simplification methods appear (in general) to be Cauchy or Lipschitz continuous (small data changes imply small semantic changes). This clearly does not hold in general for graph data. We believe the difference is due to two factors:

1. **Scivis** data is defined over *Euclidean* domains ($\mathbb{R}^m$). This allows simplification operators to readily use *continuous* Euclidean distances to *e.g.* aggregate and cluster data. An entire machinery is available for this, including basis functions and interpolation methods [14, 12].
2. All data samples have (roughly) the same importance, and the phenomenon (signal) sampled by the scivis dataset D is of bounded frequency. Hence, discarding a few samples does not affect data semantics.

In contrast:

1. **Graph** data is defined over an *abstract* graph space, whose dimensions, and even dimensionality, are not known or even properly defined. It is not always evident how to define 'proximity' between graph nodes and/or edges. There is no comparable (continuous) interpolation theory for graph data. Graph-theoretic distances are not continuous. Simply put: There is *nothing* (no information) between two nodes connected by an edge;

2. Nodes and edges can have widely different importances. There is, as we know, no similar notion of 'maximal frequency' of a graph dataset as in scivis. Hence, discarding a few samples can massively affect graph data semantics.

Image-level Simplification: A second way to handle large graph visualizations is to simplify them in the image domain. That is, given the limitations of data-space graph simplification listed earlier, rather than designing simplification operators F that act on the graph datasets, we embed the simplification into the graph rendering operator R. The key advantage here is that R acts, by definition, upon an Euclidean space (the 2D target image), where all samples (pixels) are equally important. Hence, the main proposal of image-based graph visualization is to *delay* simplification to the moment where we can reuse/adapt known scivis techniques for data simplification. Rather than first simplifying the graph data (F) and then mapping (M) and rendering (R) it, image-level techniques first map the data, and then simplify it during rendering[1]. We detail the advantages and challenges of image-based graph visualization next.
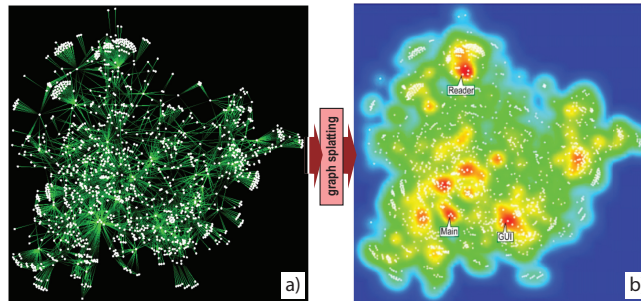
4   Image-Based Graph Visualization

Image-based graph visualization is a subfield of the larger field of image-based information visualization [20]. The name of this field can be traced back to 2002, when image-based flow visualization (IBFV) was proposed to depict large, complex, and time-dependent 2D vector fields using animated textures [60]. Key to IBFV (and its sequels) was the manipulation of the *image-space* pixels to produce the final visualization. Several advantages followed from this approach:

– *Dense* visualizations: Every target image pixel encodes a certain amount of information, thus maximizing the data-ink ratio [57];
– Clutter is avoided by construction: Rather than *scattering* dataset samples over the image space (which can lead to clutter when several such samples inadvertently overlap), samples are *gathered* and explicitly aggregated for each pixel. The aggregation function is fully controlled by the algorithm;
– Implicit *multiscale* visualizations: By simply changing the resolution of the target image (zooming in or out), users can continuously control the amount of information displayed per screen area unit;
– Exploitation of existing knowledge about image *perception* when synthesizing and/or simplifying a graph visualization;
– *Accelerated* implementations: Image-based techniques parallelize naturally over the target image pixels (much as raycasting does), so they optimally fit to modern GPU architectures [62, 25];
– Simpler *implementations vs* data-space graph simplification techniques.

---

[1] The underlying assumption here is that mapping and simplification are conceptually commutative. As discussed next, this is not always the case.

A more subtle (but present) advantage of image-based graph visualizations is their ability to reuse principles and techniques grounded in the theory and practice of image and signal processing, thereby allowing a more principled reasoning about, and control of, the resulting visualization. We next outline the main advances of this field, along with the challenges that we still see open. Given the structure of a graph in terms of nodes, edges, and attributes thereof (Sec. 2.1), we structure our discussion along the same concepts.

Fig. 3. (a) Node-link graph drawing (dataset from Fig. 2e) and (b) its graph splatting.

4.1   Node-Centric Techniques

The first image-based graph visualization, to our knowledge, is *graph splatting*, proposed in 2003 by De Leeuw and Van Liere [27]. Its intuition is simple: Given a graph drawing (layout) $M(G)$, its visualization $R(M(G))$ is the convolution of $M(G)$ with an isotropic 2D Gaussian kernel in image space. This is simply a low-pass filter that emphasizes high-density node and/or edge areas in the layout. The visualization's level-of-detail, or multiscale nature, is controlled by the filter's radius. The samples' (nodes or edges) weights can be set to reflect their importance. Figure 3(b) shows the splatting of the graph in Fig. 3(a), for the same call graph as in Fig. 2e, where the nodes' weights are set to their number of outgoing edges (fan-out factor). The resulting density map, visualized with a rainbow colormap, thus emphasizes nodes (functions) that call many other functions as red spots. This allows easily detecting such suitably-called 'hot spots' in the software system's architecture.

Graph splatting is extremely simple to implement, fast to execute (linear in the number of splatted nodes and/or edges), and easy to control by users via its kernel-radius parameter. It also forms the basis of more advanced techniques such as graph bundling (Sec. 4.2). Formally, it is a variant of the more general kernel density estimation (KDE) set of techniques used in multidimensional data analysis [49]. Its key limitation is that it assumes a good layout $M(G)$: Density hot spots appear when nodes and/or edges show up *closely* in a graph layout. So, when layout methods M place unrelated nodes close to each other, 'false positive' hot spots appear (and analogously for false negatives).

a) hand-drawn flow maps (1864)

b) Sankey diagrams (1898)

c) edge concentration (1989)

d) confluent drawings (2003)

e) flow map layouts (2005)

f) circular layouts (2005)

g) hierarchical edge bundling (2006)

h) general graph bundling (2008)

i) image-based bundling (2010)

j) dynamic graph bundling (2012)

k) CUDA universal bundling (2016)

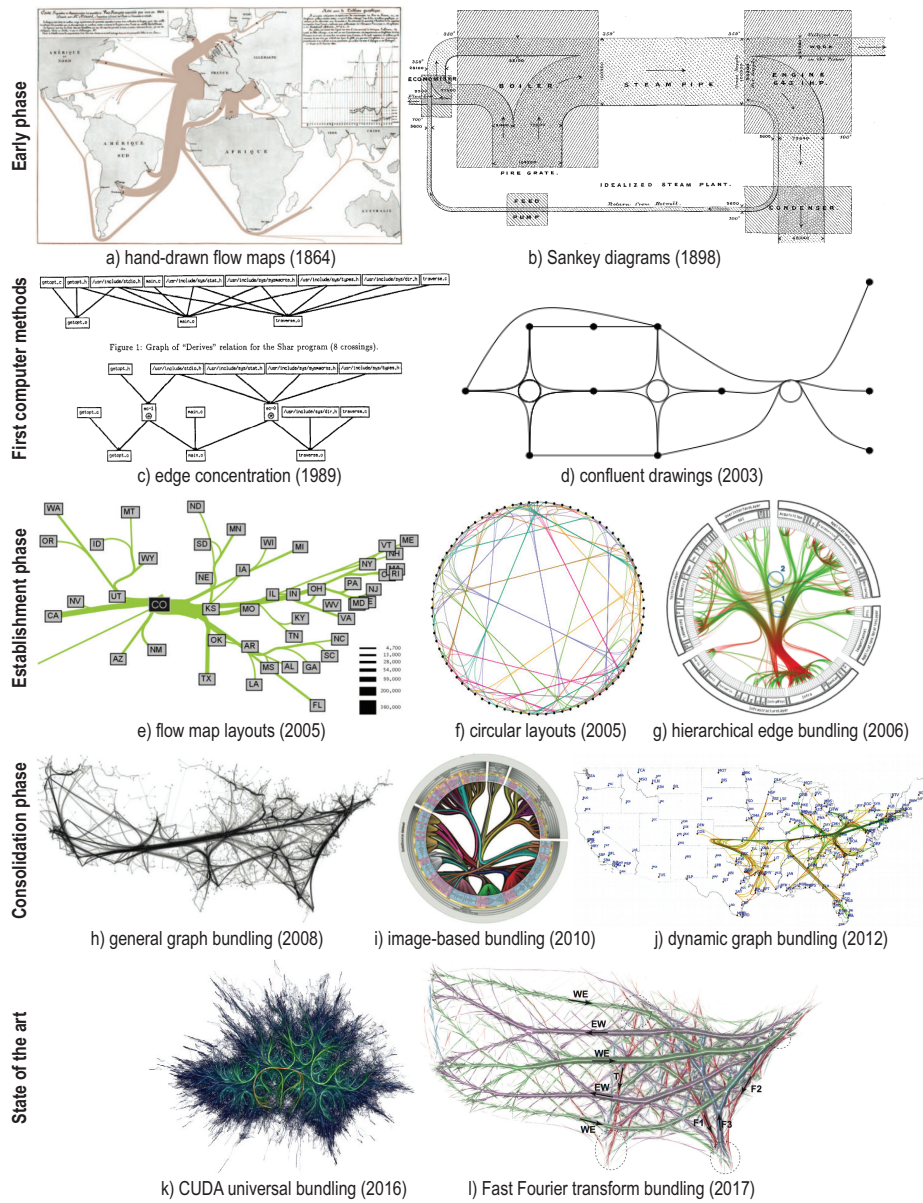l) Fast Fourier transform bundling (2017)

Fig. 4. Key moments in edge bundling history.

4.2   Edge-Centric Techniques

Graph bundling is the foremost image-based technique focusing on graph edges. Bundling has a long history (see Fig. 4 for an overview of its most important moments). We distinguish five phases, as follows (for a comprehensive recent survey, we refer to [26, 61]):

Early Phase: Minard hand-drew a so-called 'flow map' (a single-root directed acyclic graph) showing the French wine exports in 1864 [33]. While not properly a bundled graph, as no edges are grouped together, the visual style featuring curved edges whose thickness maps edge weights, suggests later bundling techniques. The design was refined in 1898 to create the so-called Sankey diagrams, which can display more complex (multiple source, cyclic) graphs;

First Computer Methods: One of the first computer-computed bundling-like visualizations was proposed by Newbery in 1989 [35]. The key novelty *vs* earlier methods is grouping edges sharing the same end nodes (so, this technique can be seen as a particular case of graph simplification by aggregation). Dickerson *et al.* coined the term 'edge bundling' in 2003 for their method that optimizes node placement and groups same-endpoints edges (via splines) to simplify graph drawings. All these techniques could handle only small graphs of tens up to hundreds of nodes and edges.

Establishment Phase: Subsequent methods focused on larger-size graphs (thousands of nodes and edges). Flow map layouts [42] generalized in 2005 the computation of Sankey-like diagrams, also first featuring the 'organic' branch-like structure to be encountered in many later techniques [9, 53, 7]. At roughly the same time (2006), two key bundling techniques emerged: Gansner *et al.* presented improved circular layouts [11], which grouped edges based on their spatial proximity in $M(G)$; Holten proposed hierarchical edge bundling [16] which grouped edges based on the graph-theoretic distance of their start and end nodes in a hierarchy of the graph's nodes. Holten also pioneered several advanced blending techniques to cope with edge overdraw (see also Sec. 4.3).

Consolidation Phase: The next phase focused on treating general graphs [18], time-dependent graphs [36], and, most importantly for our context, *image-based methods*. The latter include image-based edge bundles (IBEB [53], following the name-giving of IBFV [60]) which introduced clustering and grouped rendering of spatially close edges in the form of shaded cushions [58] to both simplify the rendered graph and emphasize distinct/crossing, bundles. IBEB reused several image-processing operators such as KDE [49], distance transforms [10], and medial axes [48] for computational speed. Next, skeleton-based edge bundles (SBEB) [9] used medial axes to actually *perform* graph bundlings, by following the simple but effective intuition that bundling a set of (close) curves means moving them towards the centerline of their hull.

State of the Art: Most recent methods focus mainly on scalability, using image-based techniques. Kernel density edge bundling (KDEEB) [21] showed that bundling a graph drawing is identical to applying mean shift, well known

in data clustering [6], on the KDE edge-density field. CUDA Universal Bundling (CUBu) [62] next accelerated KDEEB to bundle 1 million-edge graphs in sub-second time by parallelizing KDE on the GPU. Fast Fourier Transform Edge Bundling (FFTEB) [25] further accelerated CUBu by computing the KDE convolution in frequency space, thus bundling graphs of tens of millions of edges at interactive rates. As such, scalability seems to have been addressed successfully.
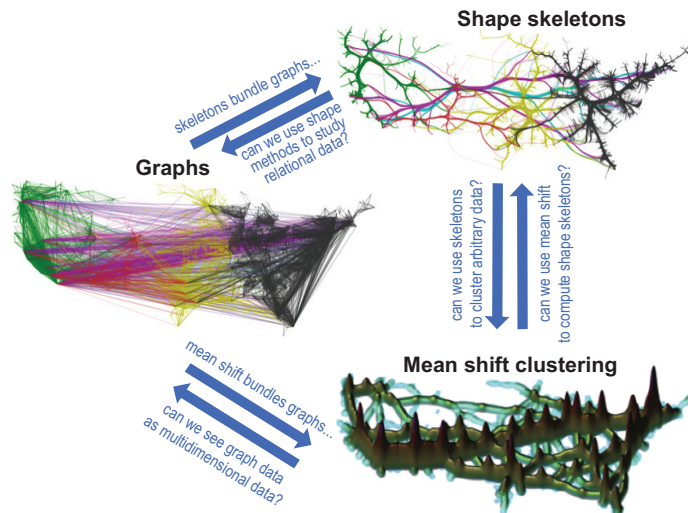
Fig. 5. Puzzling connections between graph visualization, shape analysis, and multi-dimensional data analysis.

Several points can be made about edge bundling. First, bundling is an *image-space* simplification technique of the graph *drawing* $R(M(G))$ that reduces clutter by creating whitespace between bundles, but increases overdraw (of same-bundle edges); a recent bundling formal definition as an image-processing operator is given in [26]. Image-based bundling is a *multiscale* technique, where the KDE kernel radius controls the extent over which close edges get bundled, thereby allowing users to easily and continuously specify how much they want to simplify (bundle) their graphs. Image-based methods are clearly the fastest, most scalable, bundling methods, due to the high GPU parallelization of their underlying image processing operations. Edge similarity, the bundling driving factor, can be easily defined in terms of a mix of spatial (Euclidean) and attribute-based distances [41]. More interestingly from a theoretical point, bundling exposes some puzzling connections between domains as different as data clustering [6], shape simplification [48], and graph visualization itself (Fig. 5). Briefly put:

- If skeletons can be used to bundle graphs [9], how can we further use the wealth of shape analysis methods to analyze/visualize graphs?
- If skeletons and mean shift bundle graphs [9, 21], can we use skeletons to cluster multidimensional data, or mean shift to compute shape skeletons?

– If mean shift simplifies graphs [21], could we see graphs as yet another form
  of multidimensional data?

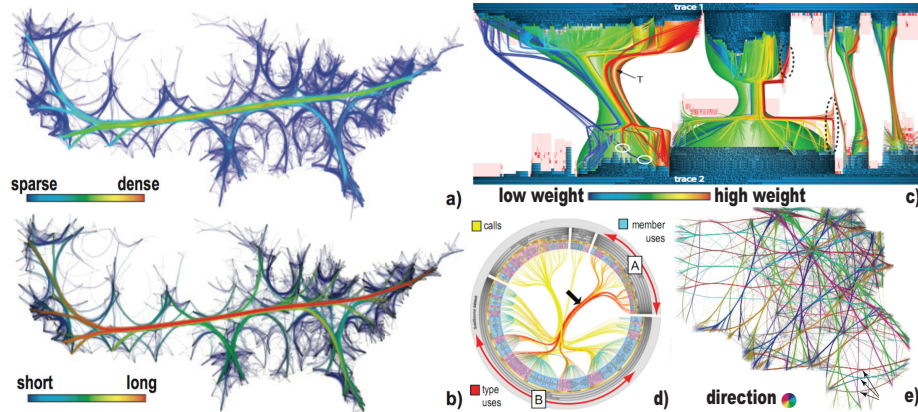These questions open, we think, a wealth of new vistas on data visualization.

Fig. 6. Attribute encoding in bundled graph visualizations.

4.3   Attribute-Centric Techniques

Graph visualization scalability also means handing high-dimensional node and/or
edge attributes (Sec. 2.2). Visualizing these is hard, since the method of choice for
handling geometric scalability – bundling – massively increases edge overdraw.
Several image-based techniques address attribute visualization, as follows.

One can directly visualize the edge-density (KDE) map *e.g.* by alpha blend-
ing [16], which is a simple form of graph splatting using a one-pixel-wide kernel.
Additionally, hue mapping can encode edge attributes, such as density [18, 62]
(Fig. 6a), length [16, 62] (Fig. 6b), quantitative weights [56] (Fig. 6c), categorical
edge types [53] (Fig. 6d), and edge directions [41] (Fig. 6e). Two main challenges
exist here. First, at edge overlap locations, attribute values e_i^j of multiple edges
$\mathbf{e}_j$ have to be *aggregated* together prior to color coding. While this is straightfor-
ward to do for *e.g.* edge density, it becomes problematic for other attributes such
as edge categorical types or edge directions. This issue parallels known challenges
in scivis (interpolation of vector fields) and infovis (aggregation of categorical
data). Secondly, there is currently no scalable method that can render *at the
same time* more than roughly two attributes per edge in high-density graph vi-
sualizations. Visualizing graphs having tens of attributes per edge ($N_E > 2$) is
an open problem. Separately, animation has been used to encode edge directions
by using particle-based techniques [19]. Interestingly, this approach resembles a
form of IBFV [60] applied to the vector field defined by the edges' tangent vec-
tors. However, in a typical vector field, the number of singularities (where IBFV
would have problems rendering a smooth, informative, animation) is quite lim-
ited; in a dense graph, this number is very high, equalling the amount of edge

crossings or, in the bundled case, overlaps of different-direction edges [9]. Hence, IBFV cannot be directly used to visualize large/complex graphs.

5   Open Challenges

Image-based techniques have shown high potential for the efficient and effective visualization of large graphs. Yet, we also see a number of key challenges that they would need to tackle to become (more) effective in practice, as follows.

Layouts: Current image-based techniques address the rendering (R) phase, but assume a suitable node layout to be given as input. As explained, computing such a layout (for large graphs) is challenging. A promising direction is to further explore analogies between dimensionality reduction (DR, used to efficiently and effectively visualize high-dimensional sample sets embedded in $\mathbb{R}^n$) and graph drawing [22]. An additional advantage of doing this is that DR can easily accommodate a wide range of similarity functions, *e.g.*, accounting for both graph structure and attributes [32]. This could open new ways to visualizing graphs having many node and/or edge attributes. Separately, it would be interesting to consider image-based bundling approaches for the *layout* of a graph's nodes.

Aggregation: Graph splatting and bundling are the techniques of choice for generating images of large graphs. However, the way in which the multiple node and/or edge attribute values that cover a given pixels are to be aggregated is currently limited to simple operations (sum, average, minimum, or maximum) [16, 62]. Such operations cannot aggregate attributes such as categorical types or edge directions. For edge directions, it is interesting to consider analogies with scivis techniques for dense tensor field interpolation [59] which address related problems. Separately, image processing has proposed a wealth of operators for detecting and emphasizing specific features present in images such as edges, lines, textures, or even more complex shapes [13]. Such operators could be readily adapted to highlight patterns of interest in image-based graph visualizations.

Quality: Measuring the quality of an (image-based) graph visualization is an open topic [37], much due to the fact that there is typically no ground truth to compare against. Still, image processing techniques can be helpful in this area, *e.g.* by providing quantitative measures for the amount of edge intersections, bends, preservation of graph-theoretic distances, or edge-angle spatial distributions, in the final image. Such image-based metrics have been successful in assessing the quality of DR scatterplot projections [31], bringing added value beyond simple aggregate metrics. Exploring their extension to graph visualizations is potentially effective. Also, such metrics could be easily used to locally *constrain* the mapping and/or rendering phases, *e.g.* to limit the amount of undesired deformations that bundling produces.

Applications: An interesting and potentially rich field for graph visualization is the exploration of deep neural networks (DNNs), currently the favored technique in machine learning. DNNs are large (millions of nodes and/or edges), attributed by several values (*e.g.* activations and weights), and time-dependent (*e.g.* during

the network training). Understanding how DNNs work, and why/where they do not work, is a major challenge in deep learning [30]. Visualizing DNNs is also very difficult, as their tightly-connected structure yields significant edge crossings and overdraw, and it is not evident how *e.g.* bundling would help for these topologies. Exploring image-based techniques for this use-case is promising.

6   Conclusion

In this paper, we surveyed current developments of image-based techniques for the visualization of large, high-dimensional, and time-dependent graphs. These techniques have major advantages – the ability of creating dense visualizations with high data-ink ratios, treatment of clutter by construction, an implicit multiscale nature able to handle large and dense graphs, and scalable implementations. We highlighted analogies and differences between image-based techniques and related techniques for the visualization of densely-sampled fields in scientific visualization. While graph data has several important differences as compared to field data, the existing similarities make us believe that existing scivis and image-processing techniques can be further adapted to further assist graph visualization. From a practical perspective, this would lead to the creation of novel efficient and effective tools for graph visual exploration. Equally important, from a theoretical perspective, this could lead to further unification of the currently still separated disciplines of scientific and information visualization.

References

1. Abello, J., van Ham, F.: Matrix zoom: A visual interface to semi-external graphs. In: Ward, M., Munzner, T. (eds.) Proc. IEEE InfoVis. pp. 127–135 (2005)
2. Archambault, D., Abello, J., Kennedy, J., Kobourov, S., Ma, K.L., Miksch, S., Muelder, C., Telea, A.: Temporal multivariate networks. In: Kerren, A., Purchase, H., Ward, M. (eds.) Multivariate Network Visualization (Proc. Dagstuhl Seminar 13201). pp. 151–173. Springer (2014)
3. Borgo, R., Kehrer, J., Chung, D.H.S., Maguire, E., Laramee, R.S., Hauser, H., Ward, M., Chen, M.: Glyph-based Visualization: Foundations, Design Guidelines, Techniques and Applications. In: Sbert, M., Szirmay-Kalos, L. (eds.) Eurographics – State of the Art Reports. The Eurographics Association (2013)
4. Brehmer, M., Munzner, T.: A multi-level typology of abstract visualization tasks. IEEE TVCG **19**(12), 2376–2385 (2013)
5. Byelas, H., Telea, A.: Visualizing multivariate attributes on software diagrams. In: Proc. IEEE CSMR. pp. 335–338 (2009)
6. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE TPAMI **24**(5), 603–619 (2002)
7. Cui, W., Zhou, H., Qu, H., Wong, P.C., Li, X.: Geometry-based edge clustering for graph visualization. IEEE TVCG **14**(6), 1277–1284 (2008)
8. Ellis, G., Dix, A.: A taxonomy of clutter reduction for information visualisation. IEEE TVCG **13**(6), 1216–1223 (2007)
9. Ersoy, O., Hurter, C., Paulovich, F., Cantareiro, G., Telea, A.: Skeleton-based edge bundles for graph visualization. IEEE TVCG **17**(2), 2364–2373 (2011)

10. Fabbri, R., da F. Costa, L., Torelli, J., Bruno, O.: 2D Euclidean distance transform algorithms: A comparative survey. ACM Computing Surveys **40**(1), 1–44 (2008)
11. Gansner, E., Koren, Y.: Improved circular layouts. In: Kaufmann, M., Wagner, D. (eds.) Proc. Graph Drawing. pp. 386–398. Springer (2006)
12. Garcke, H., Preusser, T., Rumpf, M., Telea, A., Weikard, U., van Wijk, J.J.: A continuous clustering method for vector fields. In: Moorhead, R. (ed.) Proc. IEEE Visualization. pp. 351–358 (2000)
13. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Pearson (2011)
14. Griebel, M., Preusser, T., Rumpf, M., Schweitzer, M.A., Telea, A.: Flow field clustering via algebraic multigrid. In: Proc. IEEE Visualization. pp. 35–42 (2004)
15. Herman, I., Melan con, G., Marshall, M.S.: Graph visualization and navigation in information visualization: A survey. IEEE TVCG **6**(1), 24–43 (2000)
16. Holten, D.: Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. IEEE TVCG **12**(5), 741–748 (2006)
17. Holten, D., Isenberg, P., Van Wijk, J.J., Fekete, J.D.: An extended evaluation of the readability of tapered, animated, and textured directed-edge representations in node-link graphs. In: Battista, G.D., Fekete, J.D., Qu, H. (eds.) Proc. IEEE PacificVis. pp. 195–202 (2011)
18. Holten, D., Van Wijk, J.J.: Force-directed edge bundling for graph visualization. Computer Graphics Forum **28**(3), 983–990 (2009)
19. Hurter, C., Ersoy, O., Fabrikant, S.I., Klein, T.R., Telea, A.C.: Bundled visualization of dynamic graph and trail data. IEEE TVCG **20**(8), 1141–1157 (2014)
20. Hurter, C.: Image-Based Visualization: Interactive Multidimensional Data Exploration. Morgan & Claypool Publishers (2015)
21. Hurter, C., Ersoy, O., Telea, A.: Graph bundling by kernel density estimation. Computer Graphics Forum **31**(3), 865–874 (2012)
22. Kruiger, J.F., Rauber, P.E., Martins, R.M., Kerren, A., Kobourov, S., Telea, A.C.: Graph layouts by t-SNE. Computer Graphics Forum **36**(3), 283–294 (2017)
23. Landesberger, T.V., Kuijper, A., Schreck, T., Kohlhammer, J., van Wijk, J., Fekete, J.D., Fellner, D.: Visual analysis of large graphs: State-of-the-art and future research challenges. Computer Graphics Forum **30**(6), 1719–1749 (2011)
24. Lee, B., Plaisant, C., Parr, C.S., Fekete, J.D., Henry, N.: Task taxonomy for graph visualization. In: Bertini, E., Plaisant, C., Santucci, G. (eds.) Proc. AVI BELIV. pp. 1–5. ACM (2006)
25. Lhuillier, A., Hurter, C., Telea, A.: FFTEB: Edge bundling of huge graphs by the Fast Fourier Transform. In: Seo, J., Lee, B. (eds.) Proc. IEEE PacificVis (2017)
26. Lhuillier, A., Hurter, C., Telea, A.: State of the art in edge and trail bundling techniques. Computer Graphics Forum **36**(3), 619–645 (2017)
27. van Liere, R., de Leeuw, W.: GraphSplatting: Visualizing graphs as continuous fields. IEEE TVCG **9**(2), 206–212 (2003)
28. Luebke, D.P.: A developer's survey of polygonal simplification algorithms. IEEE CG&A **21**(3), 24–35 (2001)
29. van der Maaten, L., Postma, E.: Dimensionality reduction: A comparative review (2009), tech. report TiCC TR 2009-005, Tilburg Univ., Netherlands. http://www.uvt.nl/ticc
30. Marcus, G.: Deep learning: A critical appraisal (2018), arXiv:1801.00631 [cs.AI]
31. Martins, R., Coimbra, D., Minghim, R., Telea, A.: Visual analysis of dimensionality reduction quality for parameterized projections. Computers & Graphics **41**, 26–42 (2014)

32. Martins, R.M., Kruiger, J.F., Minghim, R., Telea, A.C., Kerren, A.: MVN-Reduce: Dimensionality reduction for the visual analysis of multivariate networks. In: Kozlikova, B., Schreck, T., Wischgoll, T. (eds.) Proc. Eurographics – short papers (2017)

33. Minard, C.J.: Carte figurative et approximative des quantités de vin français exportés par mer en 1864 (1865)

34. Munzner, T.: Visualization Analysis and Design. CRC Press (2014)

35. Newbery, F.: Edge concentration: A method for clustering directed graphs. ACM SIGSOFT Software Engineering Notes **14**(7), 76–85 (1989)

36. Nguyen, Q., Eades, P., , Hong, S.H.: StreamEB: stream edge bundling. In: Didimo, W., Patrignani, M. (eds.) Proc. Graph Drawing. pp. 324–332. Springer (2012)

37. Nguyen, Q., Eades, P., Hong, S.H.: On the faithfulness of graph visualizations. In: Carpendale, S., Chen, W., Hong, S. (eds.) Proc. IEEE PacificVis (2013)

38. Nocaj, A., Ortmann, M., Brandes, U.: Untangling hairballs. In: Proc. Graph Drawing. pp. 101–112. Springer (2014)

39. Oztireli, A.C., Gross, M.: Perceptually based downscaling of images. ACM TOG **34**(4) (2015)

40. Pal, N., Pal, S.K.: A review on image segmentation techniques. Pattern Recognition **26**(9), 1277–1294 (1993)

41. Peysakhovich, V., Hurter, C., Telea, A.: Attribute-driven edge bundling for general graphs with applications in trail analysis. In: Liu, S., Scheuermann, G., Takahashi, S. (eds.) Proc. IEEE PacificVis. pp. 39–46 (2015)

42. Phan, D., Xiao, L., Yeh, R., Hanrahan, P., Winograd, T.: Flow map layout. In: Stasko, J., Ward, M. (eds.) Proc. InfoVis. pp. 219–224 (2005)

43. Post, F.H., Vrolijk, B., Hauser, H., Laramee, R., Doleisch, H.: The state of the art in flow visualisation: Feature extraction and tracking. Computer Graphics Forum **22**(4), 775–792 (2003)

44. Ropinski, T., Oeltze, S., Preim, B.: Survey of glyph-based visualization techniques for spatial multivariate medical data. Computers & Graphics **35**(2), 392–401 (2011)

45. Schaeffer, S.: Graph clustering. Computer Science Review **1**(1), 27–64 (2007)

46. Schroeder, W., Zarge, J., Lorensen, W.: Decimation of triangle meshes. In: Thomas, J.J. (ed.) Proc. ACM SIGGRAPH. pp. 65–70 (1992)

47. Schulz, H.J., Hurter, C.: Grooming the hairball-how to tidy up network visualizations? In: Proc. IEEE InfoVis (tutorials) (2013)

48. Siddiqi, K., Pizer, S.: Medial Representations: Mathematics, Algorithms and Applications. Springer (2009)

49. Silverman, B.: Density estimation for statistics and data analysis. Monographs on Statistics and Applied Probability **26** (1992)

50. Sorzano, C., Vargas, J., Pascual-Montano, A.: A survey of dimensionality reduction techniques (2014), arxiv.org/pdf/1403.2877

51. Tamassia, R.: Handbook of graph drawing and visualization. CRC Press (2013)

52. Telea, A.: Data Visualization: Principles and Practice. CRC Press (2015), 2nd edition

53. Telea, A., Ersoy, O.: Image-based edge bundles: Simplified visualization of large graphs. Computer Graphics Forum **29**(3), 543–551 (2010)

54. Telea, A., Maccari, A., Riva, C.: An open toolkit for prototyping reverse engineering visualizations. In: Ebert, D., Brunet, P., Navazzo, I. (eds.) Proc. Data Visualization (IEEE VisSym). pp. 67–75 (2002)

55. Tollis, I., Battista, G.D., Eades, P., Tamassia, R.: Graph drawing: Algorithms for the visualization of graphs. Prentice Hall (1999)

56. Trümper, J., Döllner, J., Telea, A.: Multiscale visual comparison of execution traces. In: Kagdi, H., Poshyvanyk, D., Penta, M.D. (eds.) Proc. IEEE ICPC (2013)
57. Tufte, E.R.: The Visual Display of Quantitative Information. Graphics Press (1992)
58. Van Wijk, J.J., van de Wetering, H.: Cushion treemaps: Visualization of hierarchical information. In: Wills, G., Keim, D. (eds.) Proc. IEEE InfoVis. pp. 73–82 (1999)
59. Weickert, J., Hagen, H.: Visualization and Processing of Tensor Fields. Springer (2007)
60. van Wijk, J.J.: Image based flow visualization. Proc. ACM TOG (SIGGRAPH) **21**(3), 745–754 (2002)
61. Zhou, H., Xu, P., Yuan, X., Qu, H.: Edge bundling in information visualization. Tsinghua Science and Technology **18**(2), 145–156 (2013)
62. van der Zwan, M., Codreanu, V., Telea, A.: CUBu: Universal real-time bundling for large graphs. IEEE TVCG **22**(12), 2550–2563 (2016)