Probabilistic Graph Layout for Uncertain Network Visualization

Christoph Schulz, Arlind Nocaj, Jochen Goertler, Oliver Deussen, Ulrik Brandes,
and Daniel Weiskopf, *Member, IEEE Computer Society*

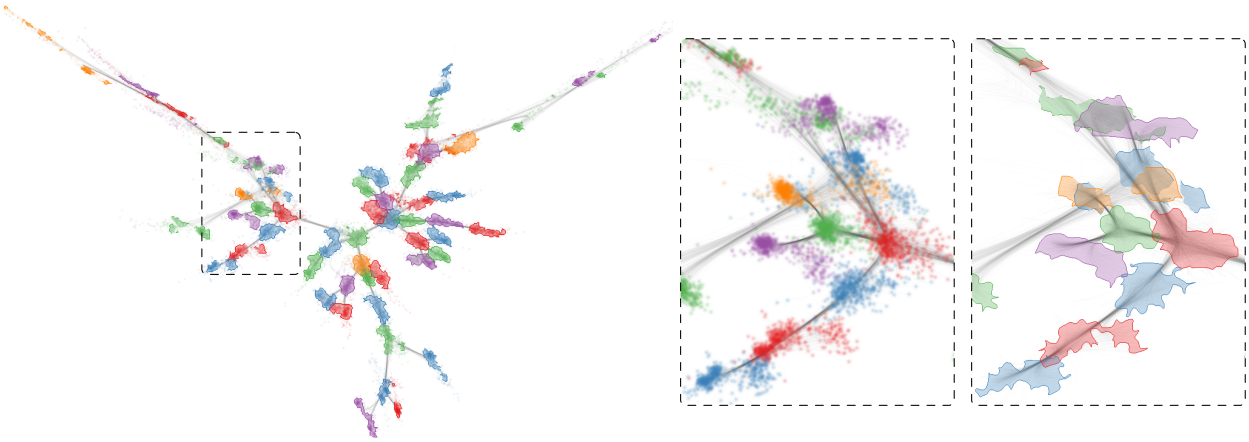


Fig. 1: Visualization of an uncertain network. Multiple samples are drawn from a probabilistic graph model and embedded in a single layout by anchoring the sampled graphs to the expected graph. The spatial distribution of sampled nodes is depicted using splatting and boundary shapes. Sampled edges are bundled using network topology. From left to right: tree overview, detail with splatted nodes, and detail with clustered shapes.

Abstract—We present a novel uncertain network visualization technique based on node-link diagrams. Nodes expand spatially in our probabilistic graph layout, depending on the underlying probability distributions of edges. The visualization is created by computing a two-dimensional graph embedding that combines samples from the probabilistic graph. A Monte Carlo process is used to decompose a probabilistic graph into its possible instances and to continue with our graph layout technique. Splatting and edge bundling are used to visualize point clouds and network topology. The results provide insights into probability distributions for the entire network—not only for individual nodes and edges. We validate our approach using three data sets that represent a wide range of network types: synthetic data, protein–protein interactions from the STRING database, and travel times extracted from Google Maps. Our approach reveals general limitations of the force-directed layout and allows the user to recognize that some nodes of the graph are at a specific position just by chance.

Index Terms—Uncertainty visualization, graph layout, graph visualization, edge bundling, Monte Carlo method.

1 Introduction

We present a technique for probabilistic graph layout and visualization, allowing visual inspection of uncertain networks and their statistical properties. Our aim is to visualize the distribution of possible realizations of a probabilistic graph that reflect certainty and uncertainty equally well. We see this as a natural extension of previous work done on uncertainty in the context of graphs, e.g. knowledge engineering and visual variables.

Various approaches have been developed to visualize *exact graphs* [40]. Yet many applications contain uncertain data due to inaccuracies, incompleteness, and inference. Typically, uncertainty is encoded as a visual variable [18] [30] into an exact graph layout. Although this is a valid strategy, we argue that the nature of a probabilistic graph is hard to understand by inspecting individual elements. The different realizations of the graph, together with their probabilities, also called *possible worlds*, are not encoded into the graph layout. Putting individual relations into context is no longer sufficient with an increasing number of nodes and edges, which further increases the layout's importance in terms of comprehension. Graph mining is bound to help, but getting visual confirmation seems appropriate when dealing with statistics. Our technique reveals statistical properties by transforming probability distributions into a two-dimensional embedding using graph layout techniques. Many *uncertain graph* models denote the existence of an edge using a probability; however, uncertainty is a fuzzy concept in visualization. Hence, we prefer the term *probabilistic graph* over *uncertain graph*. Figure 1 shows our visualization for a simple tree with probabilistic edge weights. A node is not just a single point but an entire point cloud, hence each node can occur in many regions or simply put: reflect uncertainty.

Our contribution is threefold. First, our work provides a model of probabilistic graphs. Second, we propose a formal description of the graph layout problem, along with a practical approximated solution. We present a visualization technique for probabilistic graphs that combines splatting, edge bundling, clustering, and graph coloring. At last, we demonstrate and validate our technique using representative example data sets.

2 Related Work

This work builds upon research on graph theory, graph visualization, and uncertainty visualization.

- *Christoph Schulz and Daniel Weiskopf are with VISUS, University of Stuttgart. E-mail: firstname.lastname@visus.uni-stuttgart.de.*
- *Arlind Nocaj, Jochen Goertler, Oliver Deussen, and Ulrik Brandes are with University of Konstanz. E-mail: firstname.lastname@uni-konstanz.de*

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

Graph Models: Graphs are are commonly used to represent networks composed of nodes and edges. Sometimes the existence of the relationship between two nodes is unknown due to inaccuracies, incompleteness, and inference under false assumptions [35]. For example, in biology nodes can be used to represent proteins while edges are used to represent interactions between proteins. Protein–protein interaction is an uncertainly measured or predicted process [41]. Other examples of uncertainty in graphs are the link-prediction influence [29] and obfuscated identities [4] for social networks.

The term *probabilistic graph* [26] is not to be confused with *random graph* [13]—the latter is used in conjunction with generative data models. Querying and mining *uncertain graphs* has recently received considerable attention [32]. These graphs differ from *exact graphs* in that their model expresses possible worlds instead of the actual world. Choosing good representatives that reflect the expected world can be difficult. Nevertheless, most algorithms and people work with instances of these graphs, which can be problematic [32]. Our *probabilistic graphs* can be considered one type of uncertain graph, and our approach employs probability functions instead of scalar values, leading to a powerful model. Hence, we can represent complex probabilistic processes such as dice experiments on an edge.

Graph Theory: Kobourov et al. [25] provide an overview of force-directed graph drawing algorithms. Brandes et al. [7] evaluate different types of graph distance-based drawing algorithms. Our work is based on stress majorization [17] and offline dynamic graph drawing [5]. In contrast to linking layouts over time sequentially, we align possible layouts to a reference layout using anchoring and stress majorization. Our approximated solution can be considered as a possible example of implementation. We distinguish ourselves from model estimation approaches, which may seem visually similar [20], since we propagate instead of estimating uncertainty.

One problem was to find a solution to the graph coloring problem because the number of nodes likely exceeds the number of visually distinctive colors, like noted by Ware [43]. We conservatively assume this number to be somewhere between 6 and 12 colors. Our solution is based on the ideas by Gansner et al. [16] and the Welsh-Powell algorithm [44]. We provide a discussion of how our probabilistic graph coloring differs from the classic graph coloring problem and provide a heuristic solution.

Graph and Uncertainty Visualization: For the purpose of our work, we outline overlaps and differences between graph and uncertainty visualization. We believe that structure and uncertainty of the underlying data are equally important. Hence, we use these two design dimensions to arrange related work.

The field of graph visualization is broad, as indicated by the large number of surveys for different types of graph data: Von Landesberger et al. [40] provide a classification of graphics according to their dependence and structure. Beck et al. [2] classify the depiction styles for dynamic graphs, which is helpful in evaluating possibilities for the representation of uncertain graphs. Our work adapts graph splatting [38] and hierarchical edge bundling [21]. The former is used to convey distribution of nodes and edges, whereas the latter is used to emphasize topology and keep visual clutter at a minimum. While visually similar, we distinguish ourselves from graph bundling [23], where image processing techniques are used to bundle generic graphs.

Uncertainty received broader attention in scientific visualization [8], even though uncertainty is present in information visualization affine data [1] and visual analytics [11]. For example, Feng et al. [15] propagate statistical uncertainty to parallel coordinates using a density-based approach, and Berger et al. [3] deal with uncertainty of predictions during sensitivity analysis of multivariate parameter spaces using scatter plots and parallel coordinates.

There is much research in the field of perception and awareness of uncertainty. For example, according to MacEachren et al. [30], the use of fuzziness is considered a good visual variable for uncertainty. Elmqvist et al. [36] considered the use of color for uncertainty, which we employ to depict node stress and color labels. There is an ongoing debate within the community, whether *uncertainty* is the proper term, because of cultural baggage and the fact that it tends to overemphasize uncertainty over certainty. Sacha et al. [34] argued for the concept of *trust*, which is closely related to *quality* in database science. We apply previous work on visual variables and neglect this discussion by calling our model a probabilistic graph.

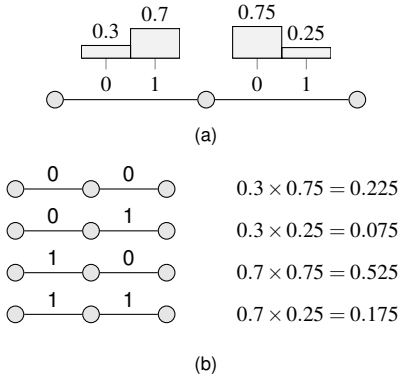


Fig. 2: Decomposition of a simple probabilistic graph (a) into all its realizations and their occurrence probabilities (b).

The combination of graph visualization and uncertainty has recently received attention. Wang et al. [42] studied the uncertainty within graph layouts. Guo et al. [18] investigated the use of visual variables to depict uncertainty of graph edges. Vehlow et al. [39] identified the concept of uncertainty for fuzzy clustering of communities. Lee et al. [28] visualized structural uncertainty in hierarchies. Our work differs in that we do not try to concentrate on local features of a graph. Instead, we transform probability distributions using graph layout techniques. During the following discussion, we distinguish between uncertainty inherent to data (probabilities) and uncertainty introduced by our visualization technique (stress and distortion) as much as possible.

3 Graph Model

Let $G^P = (V, E, F)$ be a probabilistic graph of an uncertain network with V being a set of nodes, E a set of edges, and $F = (f_{ij})_{\{i,j\} \in E}$ a set of probability density functions (PDFs). The domain of f_{ij} is a continuous random variable (weight) that maps to probability density:

$$f_{ij} : \mathbb{R} \to [0, \infty) \tag{1}$$

The PDFs are normalized:

$$\int_{-\infty}^{\infty} f_{ij}(w)\, dw = 1 \tag{2}$$

Furthermore, we assume that all probabilities are *mutually independent*, hence the joint probability density function of the entire graph is defined as:

$$f(w_1, \ldots, w_{|E|}) = \prod_{e=\{i,j\} \in E} f_e(w_e) \tag{3}$$

Based on this assumption, the area under this function must also be equal to 1:

$$\int_{-\infty}^{\infty} f(w_1, \ldots, w_{|E|})\, dw^{|E|} = 1 \tag{4}$$

Note that the independence assumption can be relaxed, as long as we have a way of sampling from the probability distribution.

For numerical computation, we discretize the continuous PDF for edge ij at weight positions $a_{ij,k} \in \mathbb{R}_0^+$, where k indexes the discrete weight. Here, we restrict ourselves to non-negative weights because typical graph layout algorithms require distance metrics, i.e., positive definiteness. In this way, we replace the continuous PDFs by discrete probability mass functions (PMFs) for a set of outcomes $A = \{a_{ij,k}\}$:

$$f_{ij} : A \to [0, 1] \tag{5}$$

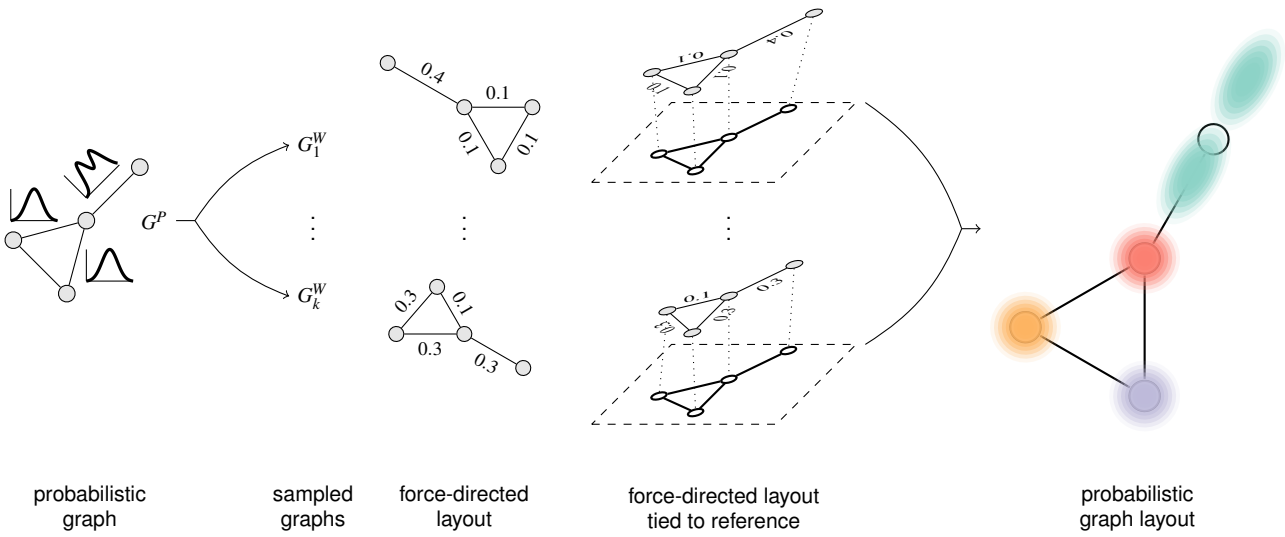


Fig. 3: Overview of the probabilistic graph layout process.

It is important to note that this definition is different from the one used by Zou et al. [45] where $A = \{0,1\}$ denotes the existence of an edge, because an edge weight of 0 is handled differently than a non-existing edge by graph layout algorithms.

Figuratively speaking, our model allows us to decompose a probabilistic graph into all possible weighted graphs. In Figure 2a, we demonstrate this concept using a simple and discrete probabilistic graph to prevent state explosion. The decomposition can be performed by sampling edge weights from each PDF, i.e., its parameter space. Exhaustive sampling results in all possible realizations of the graph, like shown in Figure 2b. Since we assume that the edge weights of the probabilistic graph are independent, we can also compute the probability for each realization. The probabilities sum up to 1, which is in line with Equation 4.

4 Graph Layout

In this section, we describe our conceptual and numerical approach to probabilistic graph layout. An exhaustive enumeration of all possible realizations of a probabilistic graph is impractical, especially with an increased number of nodes, edges, and discrete random variables. This exponential relationship between a growing number of dimensions and the required amount of samples to sufficiently represent the space is also referred to as *curse of dimensionality*.

The basic idea is to compute a graph layout using a Monte Carlo process by sampling and combining realizations to derive probability distributions of the nodes in a two-dimensional space. Hence, we aim at combining a representative subset of all possible realizations in a single layout. Simple stacking of exact graph layouts would result in confusing and unreadable results, due to ambiguities in the graph layout procedure. Furthermore, and in contrast to linear dimension reduction techniques, such as principal component analysis (PCA), graph layout algorithms are generally non-linear, which aggravates the problem. Therefore, if one considers graph layout as a projection from a high-dimensional space to $\mathbb{R}^2$, the projections must be made coherent. We suspect that there are multiple solutions to this graph drawing problem with different trade-offs.

Our approach combines a Monte Carlo method with dynamic graph drawing techniques, i.e., we extend a stress-based force-directed layout method to combine a set of possible realizations in a static visualization. Figure 3 shows the main steps of our approach:

1. Sample weighted graphs $G_1^W, \ldots, G_k^W$ from G^P by sampling the edge weights independently.
2. For each sample G_i^W: compute its node positions P_i using a force-directed layout with alignment to reference node positions P_R.
3. For each node: use its positions in $P_1, \ldots, P_k$ as approximation to its distribution in 2D space.

Similar graphs that are laid out using a force-directed method may result in similar layouts that are transformations of each other, e.g., they can be rotated or reflected. The alignment with the reference ensures coherence of sampled layouts and resolves most transformation problems that arise. We now discuss the force-directed layout and the alignment.

4.1 Force-directed Layout

The state-of-the-art approach for drawing general undirected graphs is stress minimization [17, 24], a variant of multidimensional scaling applied to graph-theoretic distances. This approach, in particular, outperforms spring embedder variants [7]. Let $G = (V,E)$ be a graph with $n = |V|$ nodes. For any pair of nodes $\{i, j\}$, with $i, j \in V$, there is an ideal distance $d_{ij} \in R^+$.

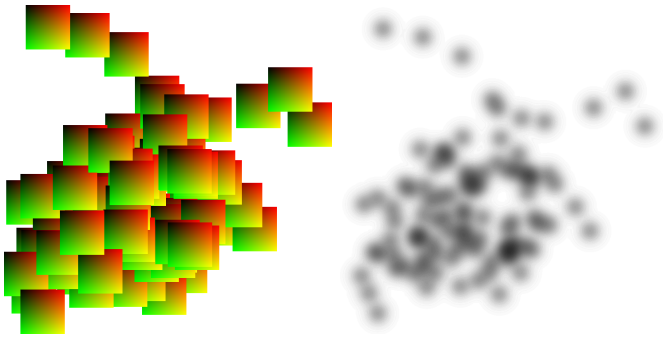
The deviation of a layout $P = (p_1, \ldots, p_n) \in \mathbb{R}^{n\times 2}$ from the ideal distances is quantified using the *stress function* [27]

$$\text{stress}(P) = \sum_{i<j} \omega_{ij} \left(||p_i - p_j|| - d_{ij} \right)^2 \tag{6}$$

where the weighting is typically chosen to be $\omega_{ij} = 1/d_{ij}^2$ to better emphasize local distances and $\|\cdot\|$ denotes the Euclidean norm. For graph drawing, the weighted lengths of the shortest paths are used for the ideal distances. Since we have a weighted graph $G = (V,E,W)$ with $w_{ij} \in W$ denoting the strength of a link, we use the inverted weight with zero mapped to infinity for the shortest path computation. Infinite distances are replaced by a distance that is 1.5 times the maximum of all pairwise finite distances within the collection of sampled networks [5]. A single node connected over such an edge will be placed far away from the main graph, but will not vanish completely to infinity. As suggested by Brandes et al. [7], we initialize the layout with PivotMDS [6] and optimize by using stress majorization [17], until we obtain a local minimum for the stress function.

4.2 Alignment with Anchoring

We want to combine layouts of multiple independent samples into one final visualization. Hence, we need to make sure that the layouts are not unnecessarily flipped or rotated. We achieve this by stabilizing the layout of each sample using anchoring [5] on a reference layout. In order to obtain such a reference layout, we first compute an *expected*



(a) (b)

Fig. 4: Node splatting: (a) For each splat, a rendering primitive is computed. (b) Afterward each splat is ray-casted and blended.

graph, G_E^W, by fixing the weight of each edge to the function's expected value $\mathrm{E}[f_{ij}]$:

$$w_{ij} = \mathrm{E}[f_{ij}] = \sum_k a_{ij,k} f_{ij}(a_{ij,k}) \tag{7}$$

We then use the positions P_E from the force-directed layout of G_E^W as the reference positions P_R for the anchoring.

The main idea of anchoring is to incorporate the reference layout into the stress function with control of its influence using a trade-off parameter. Given the layout P_i of a sampled graph G_i^W, the overall stress with respect to the stability parameter α and reference layout P_R is:

$$\mathrm{stress}(P_i; P_R, \alpha) = (1-\alpha) \cdot \mathrm{stress}(P_i) + \alpha \cdot \sum_{v \in V} \left\| p_i^v - p_R^v \right\|^2 \tag{8}$$

This affine combination allows us to do smooth blending between stacked stress-based layouts and the reference layout. If other positions such as geographic locations are given for the nodes, these can be used as reference layout, too.

5 Graph Visualization

Our visualization technique is a combination of node splatting, edge splatting and bundling, graph coloring, and density-based clustering that we will discuss in the following. We base our discussion around one question: how can we convey a distribution for each node and network topology at the same time?

5.1 Node Splatting

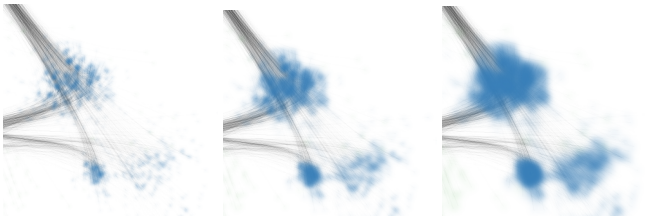
We obtain a collection of points per node in $\mathbb{R}^2$ from our Monte Carlo based graph layout. The idea is to approximate an underlying continuous distribution or scalar field for each node by applying kernel density estimation (KDE) to the Monte Carlo samples.

Our approach is based on graph splatting [38], which is essentially KDE applied to nodes. Formally, kde_n is defined by a set of samples $x_1, \ldots, x_n$, a kernel k, and a bandwidth parameter h:

$$\mathrm{kde}_n(t) = \frac{1}{nh} \sum_{j=1}^{n} k\left(\frac{t - x_j}{h}\right) \tag{9}$$

The choice of kernel k is less important because bandwidth h has more influence. The only requirement is that k is a smooth function because discontinuities would become visible as edges in the final visualization. While kernels like the Epanechnikov kernel might provide slightly better results, we choose a Gaussian kernel for its simplicity. The correct choice of kernel when facing the highly non-linear projections from graph drawing remains to be clarified.

We exploit modern graphics hardware to achieve interactive frame rates. Rendering all nodes is done in a single draw-call by ray casting and blending quads, like shown in Figure 4. The size of a quad is determined by the bandwidth parameter, while the density is evaluated per pixel.



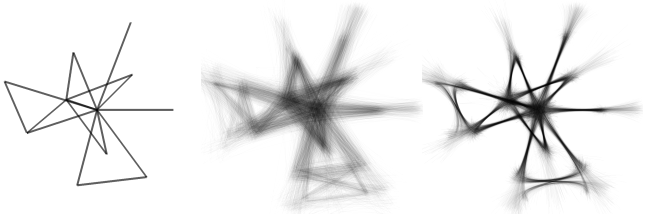
(a) (b) (c)

Fig. 5: Different node bandwidths, from undersmoothed (a) to oversmoothed (c).

The splats undersmooth or oversmooth a node distribution depending on the bandwidth h, like shown in Figure 5. An undersmoothed KDE works well, while an oversmoothed KDE seems rather flat. We presume that undersmoothed KDEs work well because stippled nodes are perceptually supported by edge lines hinting to node locations [37]. This presumption is supported by the fact that neither edges lines nor node dots work without each other.

5.2 Edge Splatting

We investigated different techniques to depict edges between node instances. Drawing one straight line between nodes hampers the mental association of nodes and edges (Figure 6a), whereas drawing straight lines between all node samples quickly leads to visual clutter concealing the topology, but depicting the distribution of edges well (Figure 6b). We believe that the network structure and mental association of node and edges should be visible at the same time. We achieve this by adopting hierarchical edge bundling [21] (Figure 6c), even though this conceals the distribution of edges. Presumably, the favored edge style depends on what aspect of the layout is of interest. In terms of hierarchical edge bundling, we define a set of edges between two sets of sampled nodes as one level of hierarchy.



(a) (b) (c)

Fig. 6: Different edge styles: (a) straight lines between centroids, (b) straight lines between all samples, (c) bundled lines between all samples.

Formally, a bundled edge is defined as rational quartic Bézier curve $C(t)$ with points $p_i \in \mathbb{R}^2$ and corresponding weights $w_i > 0$:

$$C(t, p, w) = \sum_{i=0}^{4} \binom{4}{i} t^i (1-t)^{4-i} w_i p_i \tag{10}$$

We set p_1 and p_3 to the centroids of the clusters, while p_0 and p_4 represent the source and target positions of sampled nodes, respectively. The remaining point p_3 is set to the midpoint of the segment (p_1, p_3). An example of such a quartic curve is depicted in Figure 7. The bundling strength can be controlled using the weight w_2 with other weights set to one. A high bundling strength emphasizes network structure at the cost of the visibility of the node distribution, whereas a low bundling strength emphasizes sample-to-sample connectivity.

The rendering of edges works similar to that of nodes, i.e., we splat line primitives and blend them appropriately. The main differences is that we use a thin box kernel instead of a thick Gaussian one.

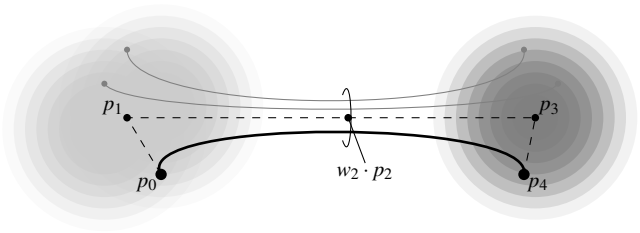


Fig. 7: Bundled edge connecting two node samples with a quartic Bézier curve. The point p_2 is the midpoint of the segment (p_1, p_3). To achieve the bundling effect, the control points are shared among all edges. The strength of the bundling can be set through the weight w_2.

5.3 Node Coloring and Labeling

Node samples are spread around their reference node, which poses a number of challenges for labeling:

Distinctness: A node's label should clearly separate a node from other nodes. We have to resort to color labels because text labeling each sample would cause visual clutter. It has been shown that the number of visually distinctive colors is limited [43]. Hence, we have a classical graph coloring problem.

Distribution: Node samples may not be spatially close to each other, i.e., there may be gaps. This will dramatically reduce the trust in the coloring, if two nodes get the same color, while being close to each other. Therefore, we have to extend the classical graph coloring problem to respect spatial proximity.

Overlap: The final layout heavily depends on how the reference layout was chosen. Hence, nodes may overlap because node distributions expand into each other. This is problem is especially pronounced if the standard deviation is bigger than than the (usually small) expected value of the PMF. If colors are blended with less than 180° hue distance, new colors will be generated [9]. This is a perceptual problem that we acknowledge and ignore for complexity reasons.

After stating the problem, we solve it approximately by reducing it to well-known problems. We compute a *country graph* [16] based on nodes and their samples, like illustrated in Figure 8. A country graph G^C consists of probabilistic nodes V^P, node samples V^W, probabilistic edges E^P, and Delaunay edges E^D [12] of the sampled points P:

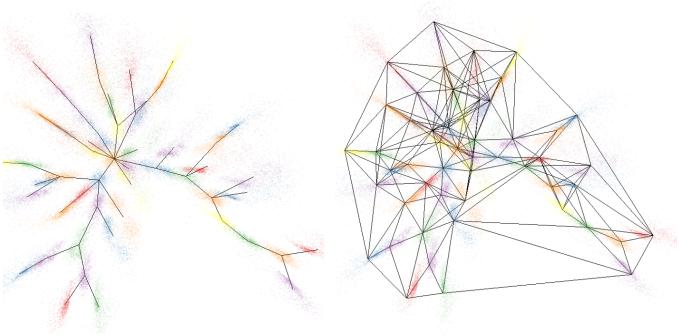
$$G^C = (V^P \cup V^W, E^G \cup E^D) \tag{11}$$

We solve the resulting graph coloring problem using the Welsh-Powell algorithm [44]. We choose this algorithm because it is easy to implement and it may be replaced by other graph coloring algorithms. The Welsh-Powell algorithm is greedy and computes the number of required colors during assignment, which we choose from a predefined color palette [19]. Colors are added using interpolation, as required. Note that we assume that the provided color palette is good enough in terms of color perception.

5.4 Clustering

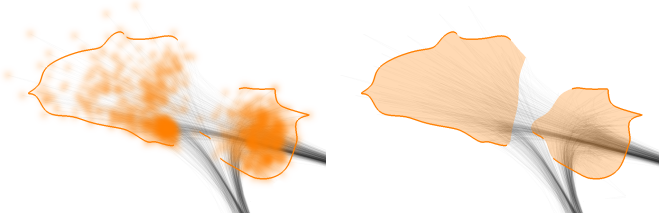
We choose to implement an auxiliary approach to analyze nodes, based on the observation that node distributions have outliers and may fall apart, i.e., into several clusters. We do not make assumptions about the PMFs, particularly it is not necessary that the PMFs are normally distributed. Arbitrary PMFs and flipping in the layout can lead to a non-coherent distribution of node positions.

Identifying *node clusters* can be a tedious task for users, which is why we provide automatic support for this process. Clustered node positions allow us to filter and abstract node distributions, like shown in Figure 9. We choose contour shapes to depict clusters because they been proven useful to group sets, e.g. for Bubble Sets [10], and color was already used. Visual scalability of our cluster shapes does not matter because the clusters are usually filtered with machine-aid to search for interesting nodes, hence the number of overlapping shapes can be kept low.



(a) (b)

Fig. 8: (a) Visualization of a tree network and its (b) proximity (or country) graph. We use the country graph to determine colors for labeling.



(a) contour and splats (b) contour and concave hull

Fig. 9: Clustering of node samples. Split clusters are visually recognized as one cluster by the half-opened contour.

The main steps of our approach are:

1. Cluster each set of node positions.
2. Optional: Filter clusters.
3. Compute a smooth concave hull of each cluster.
4. Compute a hull contour with visibility based on cluster-to-cluster geometry, inspired by *cel shading*.

We use DBSCAN [14] for clustering because it is reasonably fast and insensitive to outliers, depending on the chosen parameters. Other clustering algorithms may also suffice. Next, we calculate the Delaunay triangulation [12] to compute the convex hull of each cluster, flex edges inward to achieve the desired concavity [33]. Other algorithms such as α-shapes or marching squares may also work. The concave hull is then fitted with a B-spline and re-sampled to obtain a smooth shape. Last, we compute a contour to convey togetherness of clusters by opening the line toward other clusters of the same node—a node may have many clusters, each of which has a centroid toward which we open the line. Let c_i, c_j be centroids of a node's clusters and $\tau \in [0..1]$ a visibility threshold. The hull of cluster i consists of line segments with index k. The corresponding outward-pointing unit-length normal vectors are denoted by $\hat{n}_{i,k}$. The visibility of a line segment is then defined as:

$$\bigwedge_{j \neq i} \left(\frac{c_j - c_i}{\|c_j - c_i\|} \cdot \hat{n}_{i,k} < \tau \right) \tag{12}$$

The combination of splatting and clustering allows us to analyze the resulting layout top–down and bottom–up simultaneously. We describe this concept by example. We draw 1000 samples from a probabilistic graph. We start analysis by zooming out to get an overview and adjust the KDE bandwidth so that nodes are visible and stippled. Next, we know from experience that at 1000 samples a good proximity threshold

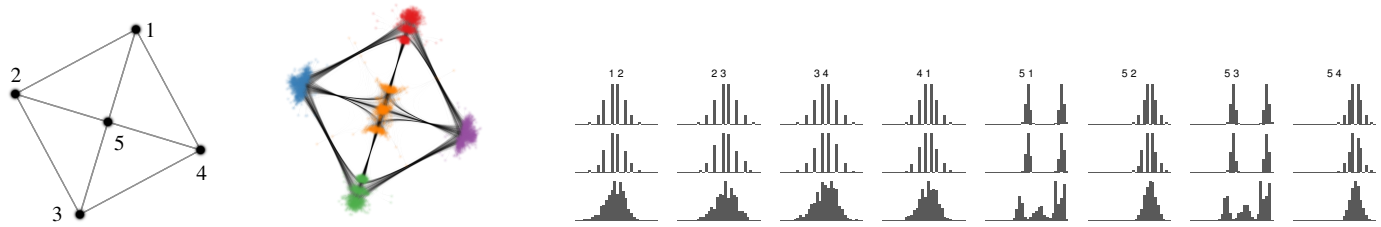


Fig. 10: Starlike graph. (a) Layout of expected graph. (b) Layout of probabilistic graph. (c) For each edge: distribution of PMF (top row), sampled edge weight (center row), and Euclidean edge length for samples (bottom row). While the different PMFs on edges are not visible in (a), they are clearly visible in (b), as the orange node is split in three pieces: one for each possible combination of the bimodal PMF on edges $(5,1)$ and $(5,3)$.

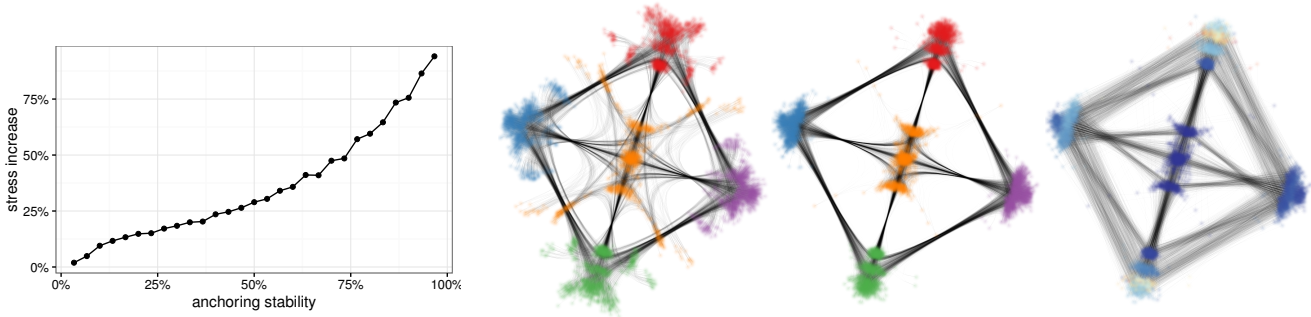


Fig. 11: Visualization of the starlike graph for various stability values α. (a) Average increase in stress for various stability values. Up to $\alpha = 0.2$, the stress is increased by at most 15%. (b) The flipping and rotation problems of the force-directed layout result in much clutter. Anchoring on the reference layout stabilizes the resulting layout (c) at $\alpha = 0.15$. The stress of each node (d) is distributed rather well; from low to high stress: blue–yellow–red.

for DBSCAN to filter noise is between 10 to 50. Hence, we only need to fine-tune the proximity radius for DBSCAN based on one typical node of choice. The resulting clustering should be fine for most nodes, therefore we can filter all nodes with less than two clusters along with corresponding shapes. Instead of the number of clusters, other criteria could be based on shape and spatial distance of clusters. The remaining nodes are potentially interesting, hence we can adjust the bandwidth for those nodes without being distracted by filtered nodes. Furthermore, we can estimate DBSCAN noise filtering parameters when rendering node splats and shapes at the same time.

6 Results

For evaluation purposes in the context of this paper, we use synthetic data, protein–protein interactions (PPI), and travel times by car.

6.1 Synthetic Data

We use synthetic graph data to get an understanding of the technique, demonstrate some effects, and gain some trust in the visualization technique.

Figure 10 shows a starlike graph consisting of 5 nodes and 8 edges. The PMFs on the edges are either unimodal or bimodal (Figure 10c, top row), but all of them have the same expected value.

The simple weighted graph layout shown in Figure 10a does not reveal anything about how distributions might interact within the network—the graph seems uniform, even though it is not. In contrast, the distributions get quite visible with our probabilistic graph layout shown in Figure 10b. For the outer nodes, it is easy to grasp whether a node is connected to unimodal or bimodal distributions by counting clusters within a node. For the center node, an interaction between two bimodal distributions is revealed, since it is thorn apart into three clusters. Note that this aggregation is related to what is called *strength* of a node for weighted graph layouts, i.e., the sum of all incoming weights. While our minimalistic example seems artificial and tailored, it demonstrates the potential of our approach and allows us to discuss validity using ground truth data, i.e., if the image still reflects the initial probabilities on edges. Graph drawing is a dimension-reduction technique, which means we inherit all of its problems. We consider the PMF of an edge well reflected if it is not distorted by our layout algorithm.

The foremost thing that affects validity is the stability value α shown in Figure 11. A very low value results in visual clutter, whereas a very high value pins the entire distribution to the expected value. We choose our example stability value by inspecting stress of our anchored layout S_α over the reference layout stress S_R, i.e., relative stress $\frac{S_\alpha}{S_R}$ introduced by anchoring. The basic idea is to get a good trade-off between having a visual reference and additional stress (Figure 11a). Increased stress means that it is more likely that distributions are not well reflected in the final layout. Also note that the discrete points get visible at low stability values (Figure 11b), because of the specified splat size. Retrospectively, there might be better techniques to reconstruct nonlinearly projected continuous distributions.

To further quantify this distortion, we compare the PMF, sampled edge weights, and the Euclidean distance between connected nodes shown in Figure 10c. Apparently, all distributions are well reflected.

Another possibility to investigate the quality of the probabilistic graph layout is to inspect the stress mapped to nodes, like shown in Figure 11d. Stress is relative, hence low does not necessarily mean good, and high does not always imply bad, but it allows us to compare two regions relatively to each other. We believe that it is important to mix relative stress and absolute function distance for validation and to gain some trust in our visualization technique. The reduction of the distortion can be made part of the optimization by choosing the right stability value, but it cannot be eliminated, since it is inherent to graph drawing.

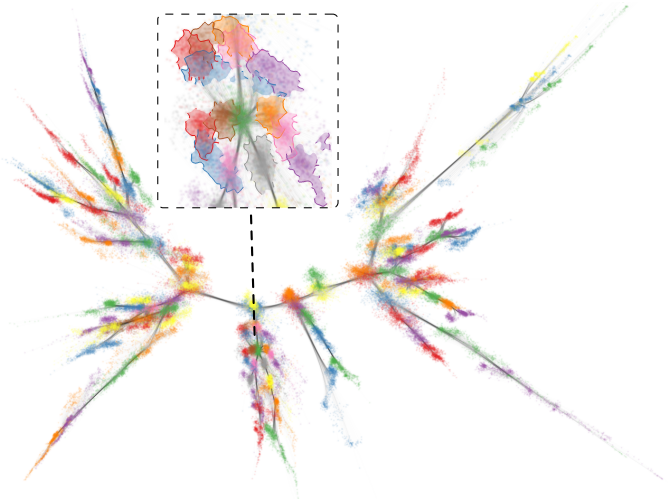


Fig. 12: Synthetic tree. The zoomed-in view shows abstracted node clusters of competing nodes, resulting in flipping.

Another interesting aspect is flipping, which requires more complex graphs to appear, like shown in Figure 12. The first aspect to notice is the strong scattering of several leaf nodes, due to the tendency toward edge weights of 0 and the high degree of freedom in graph layout, which causes the leaf nodes to flee from surrounding nodes. The detail view in Figure 12 shows a number of competing nodes. Note that node clusters (shapes) are restricted to nodes with at least two clusters. It is difficult to estimate whether nodes compete because it depends on the network topology, the reference position, and the PMFs in decreasing order of importance. For example, the distribution of the blue leaf node in the zoomed-in view of Figure 12 is divided into three major clusters. This can be disambiguated by comparing the PMF to its corresponding Euclidean distance distribution.

6.2 STRING Database

We examined data from the *STRING database*[1], which contains known and predicted protein–protein interactions. Each interaction is associated with a score $s \in [0,1]$ that is based on genetic, experimental, and literature data [41]. We interpret the score as PMF f for the interaction to actually be true, i.e.:

$$f(w) = \begin{cases} 1-s & \text{if } w = 0 \\ s & \text{if } w = 1 \end{cases} \tag{13}$$

This example is interesting for two reasons: First, two-valued distributions are at the lower end of what our technique can handle in a meaningful way. Second, it exploits zero-replacement heuristics to create clearly separated clusters, i.e., crumbled nodes are less expected to be true than coherent nodes.

Our first example is a query for *pancreatic alpha-amylase (Amy2)*. The evidence view from the STRING database website depicts the amount of information available to a set of links between the nodes (Figure 13a), whereas the expected view maps the score to value (Figure 13b). When inspecting the number of connections in the evidence view, we can tell that the number of connections does not directly correlate to the expected value, which causes some additional confusion when discovering the strong link between Amy2 and Si (Score: 0.934), whereas in our visualization these proteins are very close to each other without much of deformation. The other proteins can be grouped in descending order by score $\Delta s < 0.05$: group 1 (Pygm, Pygb, Pygl), group 2 (LOC286960, Athl1, Ctrb1), and group 3 (Cela3b, Gne, Pepf).

This order corresponds roughly to our perceived order by node coherency except for Ath1 (Figure 13c). The reason for this is that the network topology keeps Ath1 coherent, which can be confirmed by

[1] http://string-db.org/

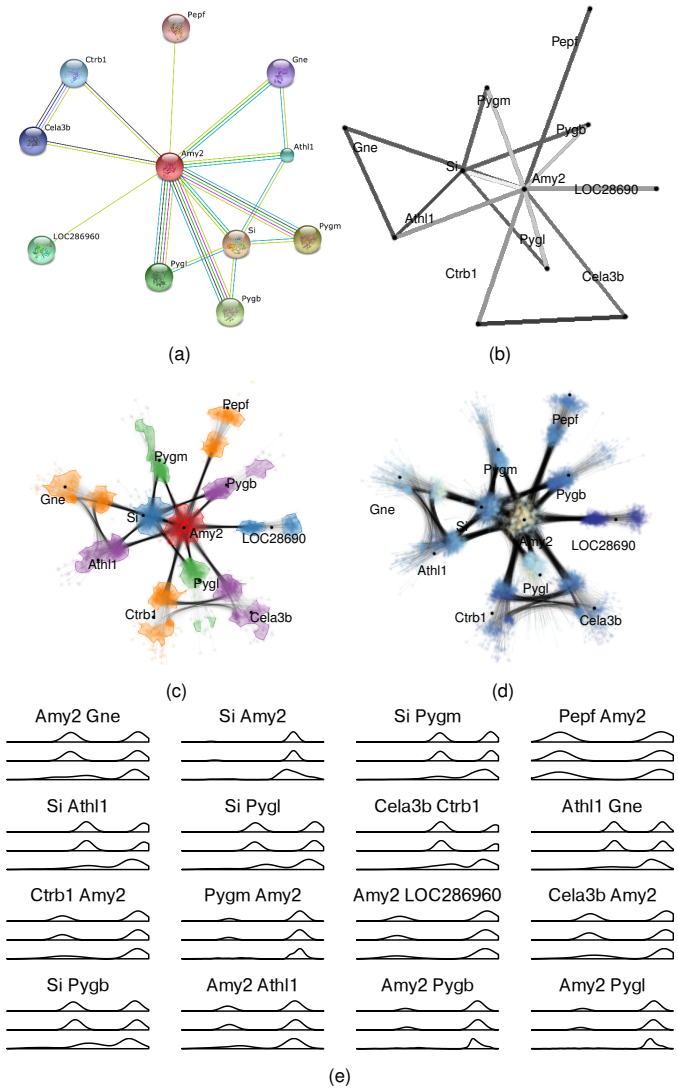
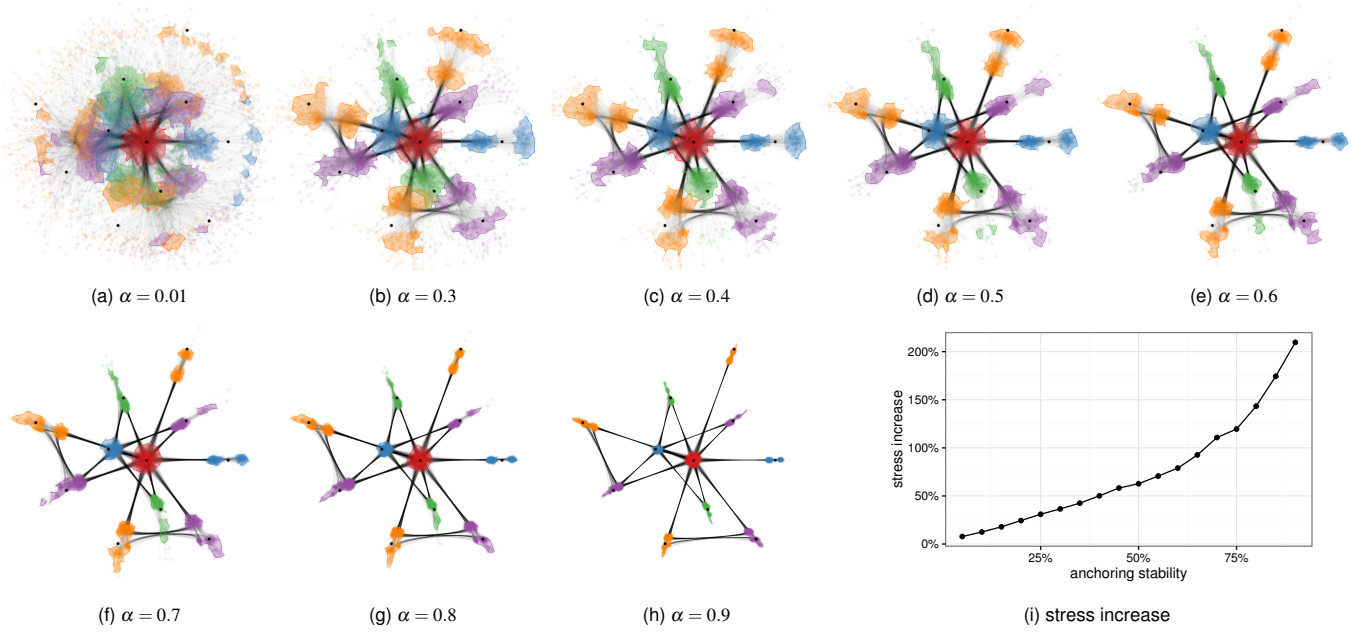


Fig. 13: Protein–protein interaction network Amy2: (a) Evidence view, (b) layout of expected graph, (c) layout using our technique, (d) stress mapped to node colors (blue: low, red: high), (e) PMFs (function plots at the top), sampled edge weights (center plots), Euclidean distances (bottom plots).

inspecting the distortion of distributions for Ath1 shown in Figure 13e, and stress mapped to nodes shown in Figure 13d. Furthermore, we can confirm that this is caused by network topology by inspecting visualizations with varying stability values shown in Figure 14. Note that orbiting of nodes around each other increases more quickly than nodes falling apart with decreasing stability and destabilizing the layout a bit aids in perceiving the network topology due to orbiting. Very low stability values increase the amount of visual clutter, whereas very high stability values match the layout of the expected graph (reference for anchoring).

Our second example is a search query for *P450 (cytochrome) oxidoreductase (POR)* with an increased search radius. Increasing the search radius inevitably creates a hairball, due to the nature of protein–protein interactions. All structure is concealed due to the sheer amount of edge crossing and coloring in the expected graph (Figure 15a). This is different with our visualization (Figure 15b), because the shape of many nodes indicate an orbit and thus structure.

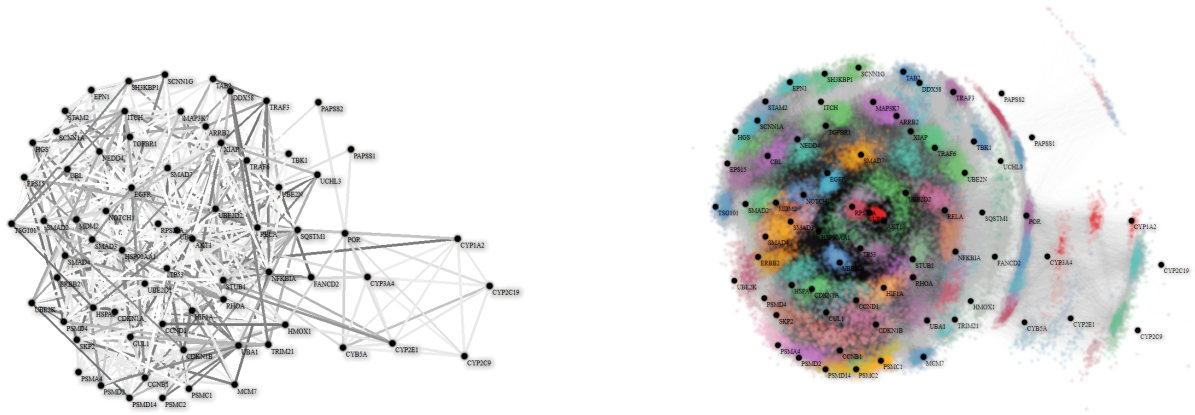
Whether the aggregation and two-dimensional representation of simple distributions is useful to biologists is a question for future work. We strongly suspect that "piling" information from edges in nodes is a



(a) $\alpha = 0.01$ (b) $\alpha = 0.3$ (c) $\alpha = 0.4$ (d) $\alpha = 0.5$ (e) $\alpha = 0.6$

(f) $\alpha = 0.7$ (g) $\alpha = 0.8$ (h) $\alpha = 0.9$ (i) stress increase

Fig. 14: Visualization of Amy2 for various stability values α. The flipping and rotation problems of the force-directed layout result in much clutter (a). Anchoring on the reference layout (black dots) with, e.g., $\alpha = 0.3$ stabilizes the layout.



(a) (b)

Fig. 15: Protein–protein interaction network POR: (a) layout of expected graph, (b) our technique illustrating the orbiting effect.

good idea because of the reduced number of visual elements (node vs. node–edge–node).

6.3 Travel Times by Car

We have experimented with travel times by car between cities. This is interesting because it is geo-referenced, non-trivial real-world data.

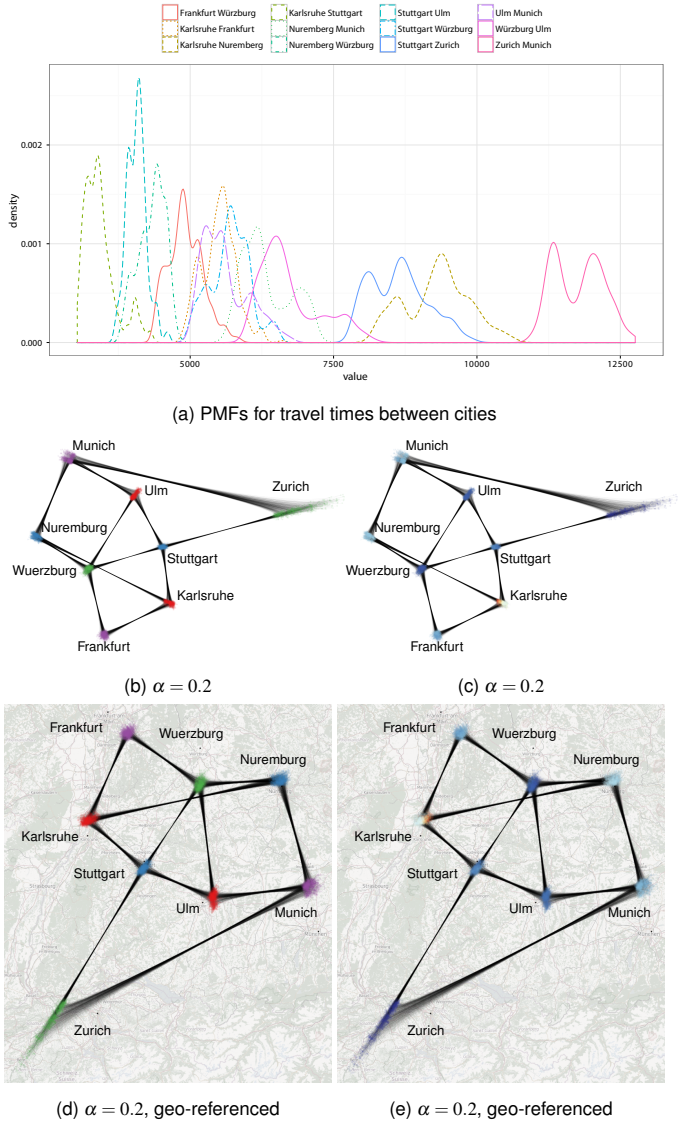
We interpret cities as nodes that are connected by edges representing major roadways. We then look at the estimated time needed to travel from one city to another. This approximated duration strongly depends on the traffic that is predicted for this road during the given time of day. For example, when traveling from city A to city B, it will make a difference if we drive this route during the rush hour or through the night, when overall traffic might be low.

We have extracted a data set of predicted travel durations between the cities using the *Google Directions API* by sampling major roadways for connected cities. Specifically, we have queried the travel time estimates for a coherent week in time intervals of one hour. The extracted network consists of eight cities located in the south of Germany and in Switzerland.

In order to get a PMF for each connection, we first build a histogram of the predicted travel times. Normalizing this histogram gives us a discrete PMF over the durations. Note that the durations are distances between two nodes. However, for our graph model we need weights, therefore we use inverted durations between two cities as weights. Note that the independence assumption is unlikely to be true—travel times are highly correlated—but we ignore this issue here.

Cities in our data set have fixed geographic locations. Hence, we can use each respective reference position for anchoring, like indicated in Section 4.2. As long as this influence is not exaggerated, the graph can still unfold its structure through the layout. Note that when using geographic positions instead of the expected graph, we have to make sure that the average geographic distance between cities is approximately the same as the average shortest path distance in the sampled graph.

Traffic data usually implies directed edges. Our initial idea was to add virtual nodes and edges to make the graph undirected, without aggregating the travel timings between two cities. Unfortunately, the resulting layout was too confusing to do visual inference on travel times. The conversion from a directed graph to an undirected graph by introducing virtual nodes and edges has too much influence on the layout. See supplemental material for more information.



(a) PMFs for travel times between cities

(b) $\alpha = 0.2$

(c) $\alpha = 0.2$

(d) $\alpha = 0.2$, geo-referenced

(e) $\alpha = 0.2$, geo-referenced

Fig. 16: (a) Travel duration by car in 8 major cities in southern Germany and Switzerland, pivoted to Karlsruhe. The probabilistic graph anchored to expected layout (b, c) and geo-locations (d, e). Note that the choice of anchor mainly affects rotation.

Therefore, we had to simplify by pivoting on one city, i.e., filtering all edges that would lead back to the pivot city. The resulting graph is non-starlike and allows us to visually infer travel times originating from one city. Figure 16 shows the sampled and filtered PMFs, the graph anchored to the expected graph (like before), and the graph anchored to the geo-locations of the cities (experimental). Note that the differences between the expected and geo-referenced layout are rather small, because anchoring with low stability values is more about directions, not about exact reference points. Hence, geo-referencing seems like a valid modification of our technique. Furthermore, the inner nodes depict a preferable direction, roughly reflecting travel times of the entire network, even though Zurich is over-exaggerated compared to Stuttgart or Karlsruhe.

7 Limitations and Challenges

We summarize the limitations of our approach and discuss the most important issues that need to be addressed. While our approach benefits from the generality of force-directed methods, it also inherits partly their weaknesses.

Layout Stability: In the force-directed layout approach, nodes can obtain completely different positions through small structural changes. Different local optima may result in reflected or rotated positions for parts of the graph. While this effect can also happen in our approach, it happens less frequently, due to the alignment with the reference layout.

Layout Ambiguity: As for the force-directed layout, it is often not clear whether the position and shape of a point cloud are due to the edge weights or due to the graph structure. Here, we see the need to incorporate more quantitative measures that would allow us to better convey the reason why a node is on this position. A simple way to do that would be to add noise on top of the incident edges of a single node, and see which influence this change has on its position.

Clearly, if the layout of a single graph sample does not convey the graph structure, e.g., if it looks like a hairball [31], then our approach is not likely to work either. Nevertheless, techniques [31] for untangling such complex structures are likely to work for our approach as well.

Computational Scalability: Our implementation of the force-directed method needs $\mathcal{O}(n \cdot (n+m) + n^2 \cdot r)$, where n is the number of nodes, m the number of edges, and r the number of iterations for stress majorization. While this only scales to graphs of medium size, other more scalable force-directed methods could be used as well [22]. The runtime to determine cluster regions for one node is on average $\mathcal{O}(k \log k)$, where k is the number of sampled graphs.

Perceptual Scalability: The more problematic limitation is in the visual distinction of the different node regions. The set of distinguishable colors is quickly exhausted, especially if the node regions are not connected. Our half-opened shapes (node clusters) allow us to search and select a limited number of distinguishable regions on top of splatting. Here, we see the need for better depiction techniques (set and distribution). In addition to that, tracing edges is hard and mapping to visual variables other than color is a big challenge. Note that the edge bundles reflect the connection between identical nodes in the graph. Thus, visual identification of single edges in such a bundle is typically not required, unless distribution and outliers of edge lines are of interest.

8 Conclusion and Future Work

We have presented a novel approach for uncertain network visualization that maps probability distributions of edges to visually perceivable splats and shapes. In this context, we have explored many aspects of filtering and depicting a set of overlapping clusters of points while maintaining the visibility of the underlying network topology.

We have applied our approach to several data sets, i.e., synthetic data, protein–protein interaction data, and travel time data with geographic reference. While it is sometimes complicated to interpret the results, due to the alternation of nodes between positions, this also allowed us to assert problems of the underlying force-directed layout method. Although not developed for this purpose, we think that our visualization approach would help analyze the impact of parameters of graph layout techniques.

Further graph layout, coloring, and visualization methods need to be developed that consider the full spatial distribution of point clouds. After examining the influence of our layout and visualization technique on various data sets, it remains to investigate its effectiveness and efficiency with controlled user experiments.

Acknowledgments

We would like to thank the German Research Foundation (DFG) for financial support within projects A01 and B02 of SFB/Transregio 161.

References

[1] C. Aggarwal and P. Yu. A survey of uncertain data algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering*, 21(5):609–623, 2009. doi: 10.1109/TKDE.2008.190

[2] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum*, 2016. doi: 10.1111/cgf.12791

[3] W. Berger, H. Piringer, P. Filzmoser, and E. Gröller. Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. *Computer Graphics Forum*, 30(3):911–920, 2011. doi: 10.1111/j.1467-8659.2011.01940.x

[4] P. Boldi, F. Bonchi, A. Gionis, and T. Tassa. Injecting uncertainty in graphs for identity obfuscation. *Proceedings of the VLDB Endowment*, 5(11):1376–1387, 2012. doi: 10.14778/2350229.2350254

[5] U. Brandes and M. Mader. A quantitative comparison of stress-minimization approaches for offline dynamic graph drawing. In *Graph Drawing, 19th International Symposium, GD 2011*, pp. 99–110. Springer, 2012. doi: 10.1007/978-3-642-25878-7_11

[6] U. Brandes and C. Pich. Eigensolver methods for progressive multidimensional scaling of large data. In *Graph Drawing, 14th International Symposium, GD 2006*, pp. 42–53. Springer, 2006. doi: 10.1007/978-3-540-70904-6_6

[7] U. Brandes and C. Pich. An experimental study on distance-based graph drawing. In *Graph Drawing, 16th International Symposium, GD 2008*, pp. 218–229. Springer, 2009. doi: 10.1007/978-3-642-00219-9_21

[8] K. Brodlie, R. Allendes Osorio, and A. Lopes. A review of uncertainty in data visualization. In J. Dill, R. Earnshaw, D. Kasik, J. Vince, and C. P. Wong, eds., *Expanding the Frontiers of Visual Analytics and Visualization*, pp. 81–109. Springer, London, 2012. doi: 10.1007/978-1-4471-2804-5_6

[9] J. Chuang, D. Weiskopf, and T. Möller. Hue-preserving color blending. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1275–1282, 2009. doi: 10.1109/TVCG.2009.150

[10] C. Collins, G. Penn, and S. Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1009–1016, 2009. doi: 10.1109/TVCG.2009.122

[11] C. Correa, Y.-H. Chan, and K.-L. Ma. A framework for uncertainty-aware visual analytics. In *IEEE Symposium on Visual Analytics Science and Technology*, pp. 51–58, 2009. doi: 10.1109/VAST.2009.5332611

[12] B. Delaunay. Sur la sphère vide. a la mémoire de georges voronoï. *Bulletin de l'Academie des Sciences de l'URSS. Classe des sciences mathematiques et na*, 7:793–800, 1934.

[13] P. Erdös and a. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959. doi: 10.2307/1999405

[14] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *Second International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, 1996. doi: 10.1.1.71.1980

[15] D. Feng, L. Kwock, Y. Lee, and R. M. Taylor II. Matching visual saliency to confidence in plots of uncertain data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):980–989, 2010. doi: 10.1109/TVCG.2010.176

[16] E. R. Gansner, Y. Hu, and S. G. Kobourov. GMap: Drawing graphs as maps. In *Graph Drawing, 17th International Symposium, GD 2009*, pp. 405–407, 2009. doi: 10.1007/978-3-642-11805-0_38

[17] E. R. Gansner, Y. Koren, and S. C. North. Graph drawing by stress majorization. In *Graph Drawing, 12th International Symposium, GD 2004*, pp. 239–250. Springer, 2004. doi: 10.1007/978-3-540-31843-9_25

[18] H. Guo, J. Huang, and D. H. Laidlaw. Representing uncertainty in graph edges: an evaluation of paired visual variables. *IEEE Transactions on Visualization and Computer Graphics*, 21(10):1173–1186, 2015. doi: 10.1109/TVCG.2015.2424872

[19] M. Harrower and C. A. Brewer. ColorBrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003. doi: 10.1179/000870403235002042

[20] P. D. Hoff, A. E. Raftery, and M. S. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97(460):1090–1098, 2002. doi: 10.1198/016214502388618906

[21] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006. doi: 10.1109/TVCG.2006.147

[22] Y. Hu. Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71, 2005.

[23] C. Hurter, O. Ersoy, and A. Telea. Graph bundling by kernel density estimation. *Computer Graphics Forum*, 31(3):865–874, 2012. doi: 10.1111/j.1467-8659.2012.03079.x

[24] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989. doi: 10.1016/0020-0190(89)90102-6

[25] S. Kobourov. Force-directed drawing algorithms. In R. Tamassia, ed., *Handbook of Graph Drawing and Visualization*, pp. 383–408. CRC Press, 2013.

[26] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques – Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[27] J. B. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, 1964.

[28] B. Lee, G. G. Robertson, M. Czerwinski, and C. S. Parr. CandidTree: Visualizing structural uncertainty in similar hierarchies. In *Human-Computer Interaction*, INTERACT'07, pp. 250–263. Springer, 2007. doi: 10.1007/978-3-540-74800-7_20

[29] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58:1019–1031, 2007. doi: 10.1002/asi

[30] A. M. MacEachren, R. E. Roth, J. O'Brien, B. Li, D. Swingley, and M. Gahegan. Visual semiotics & uncertainty visualization: An empirical study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2496–2505, 2012. doi: 10.1109/TVCG.2012.279

[31] A. Nocaj, M. Ortmann, and U. Brandes. Untangling the hairballs of multi-centered, small-world online social media networks. *Journal of Graph Algorithms and Applications*, 19(2):595–618, 2015. doi: 10.7155/jgaa.00370

[32] P. Parchas, F. Gullo, D. Papadias, and F. Bonchi. Uncertain graph processing through representative instances. *ACM Transactions on Database Systems*, 40(3):1–39, 2015. doi: 10.1145/2818182

[33] E. Rosén, E. Jansson, and M. Brundin. Implementation of a fast and efficient concave hull algorithm. Project report, University of Uppsala, 2014.

[34] D. Sacha, H. Senaratne, B. C. Kwon, G. Ellis, and D. A. Keim. The role of uncertainty, awareness, and trust in visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):240–249, 2016. doi: 10.1109/TVCG.2015.2467591

[35] M. Skeels, B. Lee, G. Smith, and G. Robertson. Revealing uncertainty for information visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pp. 376–379. ACM, 2008. doi: 10.1145/1385569.1385637

[36] S. Tak and A. Toet. Color and uncertainty: It is not always black and white. *EuroVis – Short Papers*, 2014. doi: 10.2312/eurovisshort.20141157

[37] M. Tory, C. Swindells, and R. Dreezer. Comparing dot and landscape spatializations for visual memory differences. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1033–1039, 2009. doi: 10.1109/TVCG.2009.127

[38] R. Van Liere and W. De Leeuw. GraphSplatting: Visualizing graphs as continuous fields. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):206–212, 2003. doi: 10.1109/TVCG.2003.1196007

[39] C. Vehlow, T. Reinhardt, and D. Weiskopf. Visualizing fuzzy overlapping communities in networks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2486–2495, 2013. doi: 10.1109/TVCG.2013.232

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

[41] C. von Mering, L. J. Jensen, B. Snel, S. D. Hooper, M. Krupp, M. Foglierini, N. Jouffre, M. A. Huynen, and P. Bork. STRING: Known and predicted protein-protein associations, integrated and transferred across organisms. *Nucleic Acids Research*, 33:D433–7, 2005. doi: 10.1093/nar/gki005

[42] Y. Wang, Q. Shen, D. Archambault, Z. Zhou, M. Zhu, S. Yang, and H. Qu. AmbiguityVis: Visualization of ambiguity in graph layouts. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):359–368, 2016. doi: 10.1109/TVCG.2015.2467691

[43] C. Ware. Foundations for an applied science of data visualization. In *Information Visualization*, pp. 1–30. Elsevier, 2013. doi: 10.1016/B978-0-12-381464-7.00001-6

[44] D. J. A. Welsh and M. B. Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86, 1967. doi: 10.1093/comjnl/10.1.85

[45] Z. Zou, J. Li, H. Gao, and S. Zhang. Mining frequent subgraph patterns from uncertain graph data. *IEEE Transactions on Knowledge and Data Engineering*, 22(9):1203–1218, 2010. doi: 10.1109/TKDE.2010.80