Erschienen in: IEEE Transactions on Visualization and Computer Graphics. IEEE. 2025, 31(1), S. 1039-1049
https://doi.org/10.1109/tvcg.2024.3456312

Quality Metrics and Reordering Strategies for Revealing Patterns in BioFabric Visualizations

Johannes Fuchs, Alexander Frings, Maria-Viktoria Heinle, Daniel A. Keim, Sara Di Bartolomeo

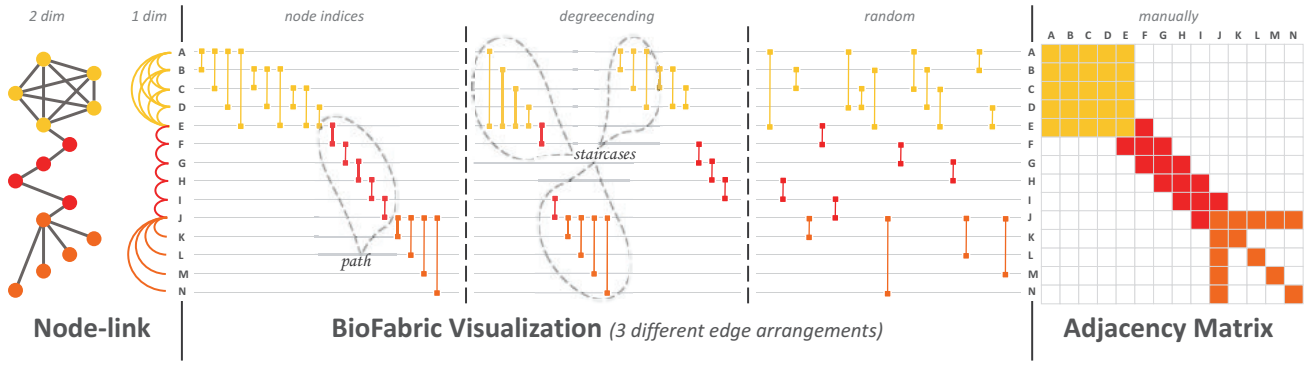


Fig. 1: The same synthetic data is visualized on a node-link diagram, BioFabric, and adjacency matrix. The color of the entities reflects the respective pattern (■ ≙ *clique*, ■ ≙ *path*, ■ ≙ *fan*). The edge order in BioFabric has a huge influence on the appearance of patterns. A random edge order shows no topological structure, whereas our *degreecending* technique reveals three *staircases* and one *path*.

Abstract—Visualizing relational data is crucial for understanding complex connections between entities in social networks, political affiliations, or biological interactions. Well-known representations like node-link diagrams and adjacency matrices offer valuable insights, but their effectiveness relies on the ability to identify patterns in the underlying topological structure. Reordering strategies and layout algorithms play a vital role in the visualization process since the arrangement of nodes, edges, or cells influences the visibility of these patterns. The BioFabric visualization combines elements of node-link diagrams and adjacency matrices, leveraging the strengths of both, the visual clarity of node-link diagrams and the tabular organization of adjacency matrices. A unique characteristic of BioFabric is the possibility to reorder nodes and edges separately. This raises the question of which combination of layout algorithms best reveals certain patterns. In this paper, we discuss patterns and anti-patterns in BioFabric, such as staircases or escalators, relate them to already established patterns, and propose metrics to evaluate their quality. Based on these quality metrics, we compared combinations of well-established reordering techniques applied to BioFabric with a well-known benchmark data set. Our experiments indicate that the edge order has a stronger influence on revealing patterns than the node layout. The results show that the best combination for revealing staircases is a barycentric node layout, together with an edge order based on node indices and length. Our research contributes a first building block for many promising future research directions, which we also share and discuss. A free copy of this paper and all supplemental materials are available at `https://osf.io/9mt8r/?view_only=b70dfbe550e3404f83059afdc60184c6`.

Index Terms—Network Visualization, Graph Drawing, Graph Layout Algorithms, BioFabric, Graph Motif

1 Introduction

Node-link diagrams or adjacency matrices are well-known visualizations for analyzing relational data like social networks [1] or biological interactions [2]. Their effectiveness in revealing patterns in the data relies on the arrangement of nodes, edges, or cells. The BioFabric visualization, pioneered by Longabaugh [39], combines node-link diagrams and adjacency matrices elements. It offers a unique layout where nodes are depicted as rows on a grid, and edges are represented as vertical lines spanning these rows.

Compared to node-link diagrams and adjacency matrices, BioFabric comes with some disadvantages. The tabular structure of the visualization does not allow for the flexible positioning of nodes and edges as in node-link diagrams. Due to the layout of the edges, it is not as scalable as adjacency matrices [26], does not support path-tracing tasks well, and is supposed to be less effective in conveying neighbors or clusters compared to adjacency matrices [42].

On the positive side, even in highly connected networks, edges do not overlap with each other, avoiding cluttered "hairballs" that might occur in node-link diagrams. Another advantage is the possibility of displaying multivariate data adequately by either embedding or overlaying the information on the visual markers directly or displaying the information separately in a juxtaposed view [26]. The tabular structure enables the sorting of vertices, like in adjacency matrices [43], but also edges according to some data characteristics [42].

This ability to independently arrange rows and columns caught our attention as a graph visualization problem. The challenge of positioning network elements is indeed central to graph drawing. Arranging rows and columns in BioFabric is similar to the sorting strategies for adjacency matrix visualizations [5] and can also be likened to graph layout algorithms used in node-link visualizations. In both of these network visualization approaches, reordering strategies and layout methods aim to enhance readability based on several quality criteria. Such criteria include, for example, minimizing the number of crossings in node-link diagrams [45, 51] or highlighting patterns in adjacency matrix visualizations [6]. Given a certain analysis task, a different visualization technique might be preferable. While adjacency matrices are particu-

- *Johannes Fuchs is with the University of Konstanz. E-mail: fuchs@dbvis.inf.uni-konstanz.de.*
- *Alexander Frings is with the University of Konstanz.*
- *Maria-Viktoria Heinle is with the University of Konstanz.*
- *Daniel A. Keim is with the University of Konstanz.*
- *Sara Di Bartolomeo is with the University of Konstanz and TU Wien.*

larly effective at communicating *cliques*, node-link visualizations excel in *path* tracing [43]. The overall layout of Biofabric presents unique possibilities in what user tasks could be supported best.

In this paper, we are particularly interested in exploring reordering strategies to optimize for the appearance of certain patterns, which is a common problem to explore in both node-link diagrams and adjacency matrices. Therefore, we discuss and transfer well-established layout algorithms to BioFabric, combine them in a meaningful way, and finally evaluate their effectiveness in revealing specific patterns, which we called *staircases*, *escalators*, and *runways*.

We contribute:

- A thorough **examination of pattern formation** within Biofabric visualizations, underscoring the unique advantages of our approach.
- The development of **quality metrics** aimed at quantifying the visibility and coherence of these patterns.
- The **presentation and implementation of various reordering strategies**, formulated to enhance pattern recognition.
- A detailed **experiment** that not only tests various combinations of node and edge orderings but also quantifies the occurrence and quality of detected patterns.

We believe that focusing on pattern optimization in Biofabric could lead to a host of new and fascinating questions and challenges in the area of graph drawing. This opens up discussions on several important topics:

- Which patterns should we aim to improve in Biofabric visualizations? Is there more that could stand out in addition to the ones discussed in this paper?
- What kind of combined approaches for rearranging nodes and edges work best for bringing out these patterns?

While our paper begins to explore some of these questions, we acknowledge that fully answering them represents an exciting avenue for future work. Although we delve into these topics to a degree, we believe that they warrant deeper exploration to truly harness their potential and encourage other researchers to take up on these associated challenges.

All of our supplemental material, implementation, and experiment results can be found on osf.io[1]. Additionally, an online prototype showcasing our research is available here: `https://biofabric.dbvis.de/`. Throughout the paper, we use color coding to facilitate the understanding of interrelated patterns and motifs:

- indicates the presence of *fans*, *stars*, *staircases*, and *runways*;
- encodes *cliques*, *communities*, and *blocks*;
- highlights *paths* and *bands*;
- represents temporal patterns like *trends*, *shifts*, and *periodicities*;
- displays noise like *bandwidth*, *escalators*, and *podiums*.

2 Background

Network motifs are recurring sub-graphs within a network or across networks, crucial for analyzing topological structures. They can be identified automatically [15, 32, 41] or visually explored by analyzing patterns in information visualizations. Real-world graphs often contain overlapping patterns across various scales. As a result, the described visual patterns may not always be distinctly discernible and can appear merged in practice. Therefore, an edge might contribute to two different patterns (i.e., the edge connecting the vertices "J" and "I" in Figure 1 contributes to the *path* but also to the *star*).

The BioFabric (BF) visualization technique is highly related to one-dimensional node-link diagrams (NLD) because of the linear node layout, adjacency matrices (AM) due to the similar tabular arrangement, and massive sequence views (MSV) with their identical visual encoding. Since the representation of patterns varies across different visualizations, we focus on the appearance and exposure of these patterns.

2.1 Node-link Diagram (NLD)

In two-dimensional NLDs, vertices can be positioned without any restrictions on the canvas. In contrast, vertices are bound on a single axis in one-dimensional arrangements like in arc diagrams [52], circular layouts [28], or, more generally speaking, book embeddings [33].

[1] `https://osf.io/9mt8r/?view_only=b70dfbe550e3404f83059afdc60184c6`

Independent of the aforementioned dimensionality, the overall goal is to find a layout that reveals certain patterns to understand the underlying network structure better. For example, force-directed algorithms [25] position closely related vertices next to each other, whereas non-adjacent entities should be separated [9]. As a result, these algorithms try to combine nodes into *cliques* or *communities*. Alternative layouts should be preferred if different structural features are more important. Other approaches focus on quality metrics like minimizing edge crossings [18] or centrality measures like node degree [8] for positioning vertices.

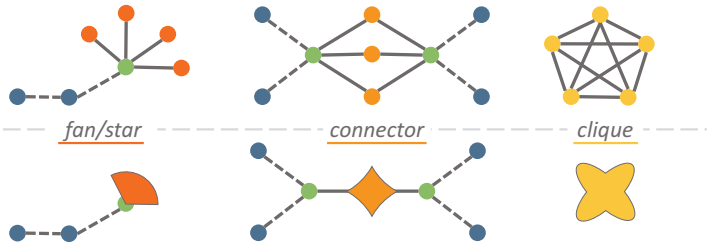


Fig. 2: **Motifs:** Simplification of patterns in NLDs [23].

To further enhance pattern detection, Dunne et al. [23] introduced three motif simplifications to summarize individual nodes in glyph-like representations (Figure 2). A *fan* motif, also known as a *star* pattern, contains an inner node connected to multiple leafs. A *connector* motif summarizes mediator nodes between groups of vertices. A *clique* motif comprises vertices with pairwise connections between all clique members.

Fig. 3: **Thread Arcs:** Visible patterns in e-mail communication [36].

A famous example of one-dimensional NLDs is arc diagrams. Initially, they were introduced to show repetitions in strings like sheet music or DNA sequences where the order of nodes is implicitly dictated by the underlying data [52]. The software "Thread Arcs" [36] uses a similar visualization to explore e-mail communication but additionally offers ordering strategies (i.e., chronological or hierarchical) to reveal patterns such as *bushy* or *narrow* (Figure 3). A hierarchical order of vertices is also applied in the circular arrangement from Holten [35]. However, there are also approaches that exploit quality metrics like minimizing edge length [28] or edge crossings [47] or centrality measures like the degree [46] to reorder nodes.

2.2 Adjacency Matrix (AM)

Each graph motif corresponds to a visual pattern in an AM visualization. We took the categorization from Behrisch et al. [5, 7] to structure different algorithms and patterns. They introduced four patterns and two anti-patterns essential for data analysis (Figure 4).

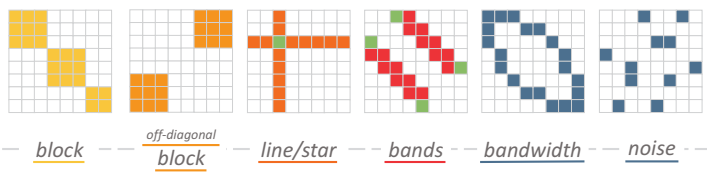


Fig. 4: Literature agrees on six **discernible patterns** in AMs [5].

The *block* pattern corresponds to a *clique* or *community* in NLDs. It appears in ordered AMs whenever strongly connected clusters are present in the underlying topology. There are many different reordering approaches focusing on revealing *block* structures. Robinsonian approaches optimize for similarities and dissimilarities between rows and columns. One

such algorithm, Heuristic Simulated Annealing [10], seeks to make *block* patterns visible by employing heuristic optimization. Similarly, biclustering approaches, such as the probabilistic plaid model [48], perform clustering on rows and columns to enable the identification of overlapping biclusters and coherent *block* patterns. When focusing more on high-level structures than fine patterns, dimension reduction techniques like PivotMDS [44] should be preferred. If performance is an issue, heuristic approaches, like Barycentric [27, 40], are a suitable choice. They enhance computational efficiency while strategically positioning clusters.

Whereas *block* patterns represent *cliques*, *off-diagonal block* patterns act as *mediators* or *connectors* of *cliques*. Research suggests applying spectral matrix reordering methods to reveal those kinds of patterns [5]. An example is the Rank-two Ellipse Seriation [11] that leverages eigenvalues to extract core matrix structures with dominant dimensions.

Line or *star* patterns depict continuous horizontal and vertical lines, indicating a vertex's strong connections to multiple distinct vertices. This pattern aids analysts in understanding and reasoning about the overall connectivity within the network since the line length corresponds to the node degree, providing valuable insights into a network structure. Magnostics is an approach to depict such patterns with its "profile descriptor" automatically [6].

A special case of *line* patterns are *bands*. They reflect continuous off-diagonal lines that refer to *paths* in a network. Unlike *lines*, they show vertices with relatively few connections to other vertices and are especially helpful when adjacency relationships and connectivity are the focus of analysis. Graph theoretic approaches, such as Reverse Cuthill-McKee (RCM) [31], leverage graph structures to optimize linear orders based on graph-theoretic layout cost functions. As a result, vertices with similar connection patterns are positioned together, potentially revealing *band* patterns. However, since those algorithms usually reduce the bandwidth, *bandwidth* anti-patterns might emerge. The *bandwidth* pattern contributes minimally to interpretation if the inner part of the bandwidth enclosure lacks structure. Similarly, *noise* occurs when ordering fails to reveal the underlying graph topology or when no structure exists. This is also called salt-and-pepper and represents the classic anti-pattern for an AM.

2.3 Massive Sequence View (MSV)

From a visual encoding perspective, MSVs are the most related to BF. They are used to visualize dynamic networks representing temporal changes along the x-axis. Since the position of edges is fixed due to their temporal location, reordering strategies focus solely on arranging nodes. As a result, different reordering strategies and, consequently, patterns emerge. Elzen et al. [50] differentiate between two different kinds of patterns (i.e., temporal patterns and structural patterns) (Figure 5).

Fig. 5: **Temporal patterns:** In MSVs the x-axis encodes time. Therefore, patterns are partially different compared to NLDs and AMs [50].

Temporal patterns comprise *trends*, *counter trends*, *shifts*, *periodicities*, and *anomalies*. *Trends* and *counter trends* show increases or decreases in transaction frequencies by blocks of varying sizes. In contrast, *periodicities* represent recurring behavior identifiable by evenly spaced sequential blocks of consistent size. *Shifts* become apparent when a *periodic* pattern experiences an abrupt or gradual disruption. If the *shift* occurs between two homogeneous transaction patterns, it is called *anomaly*. To reveal those temporal patterns, nodes can be arranged according to their temporal activity [50]. Nodes with most transactions at the beginning are positioned near the top, whereas nodes communicating most at the end are layouted near the bottom. Nodes communicating homogeneously are arranged close to the center.

Structural patterns (i.e., *community* or *star*) share characteristics similar to those of already known patterns. Like the *block* patterns in AMs or *cliques* in NLDs, *communities* consist of multiple nodes with many internal transactions representing a block-like structure in the visualization. Similar to the force-directed layout in NLDs, the application TimeArcs positions highly related nodes close to each other [17]. The recursive layout algorithm "recurrent neighbors" tries to identify *communities* by taking the degree of nodes as a quality measure and positioning adjacent nodes next to each other with the top talkers close to the center [38]. "Community-based node ordering" follows a similar strategy by keeping nodes with many transactions between themselves close [37]. Another approach focuses on minimizing the overall edge length with "simulated annealing" taking the standard deviation of the edge lengths, the block overlap, or a combination of the two as cost function [50]. *Star* patterns are similar to the corresponding patterns in NLDs or AMs. They show single, well-connected vertices. Ordering according to the degree, like in the "attribute order" strategy from Elzen et al., should highlight those *star* structures [50].

3 Patterns in BioFabric

Patterns in BF have received only a little research attention. The original paper introducing BF only mentions the *sawtooth* pattern [39] that directly corresponds to the *star* pattern in NLDs and MSVs, or *line* pattern in AMs. Since MSVs share a similar visual encoding, most patterns will look alike. The *community* pattern, for example, is directly transferable to BF. However, temporal patterns cannot be transferred since the x-axis in BF does not convey a chronological order.

To explore BF's pattern space, we first transfer already-known patterns and investigate their appearance and variations while linking them to data characteristics. The Gestalt Laws *proximity*, *similarity*, and *continuity* are the biggest drivers for perceiving these visual structures. Second, we copy patterns from NLDs and AMs to BF and discuss their perceptibility. For each pattern, we identify positive and negative properties and have a brief discussion about quality metrics to evaluate how well-formed the structures are in the layout. To recognize patterns better, we use a series of walkway-related metaphors to describe them.

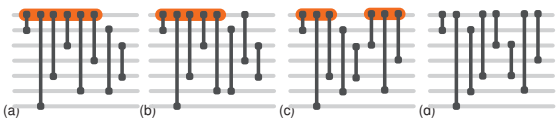
3.1 The Runway

The *runway* describes a generalized *sawtooth* pattern and looks like a single straight horizontal line (Figure 6). It appears when many connections of a single node are positioned next to each other without having a specific order. Long *runways* show well-connected nodes in the network and are, therefore, related to *star* or *line* patterns.

To reveal *runway* patterns, all connections of a single node must be next to each other. The pattern is independent of node order; however, ordering nodes by degree will reveal optimal *runways*. To evaluate the quality of a *runway*, we prefer fewer longer *runways* than many shorter ones. To quantify this, we consider the ratio of connections within each *runway* compared to the respective node degree. This calculation is performed for all *runways*. The final quality metric $q_{runways}$ is obtained by summing up all ratios and dividing by the number of runways (Equation (1)).

$$q_{runways} = \frac{\sum^{runways} (length/degree)^2}{\#runways} \tag{1}$$

Figure 6 exemplifies different runway configurations and their respective quality. An optimal *runway* results in the highest quality (Figure 6a), where all the edges associated with the topmost node are aligned on one line, highlighting the degree of the node. If the pattern is missing connections (Figure 6b) or is split (Figure 6c), quality drops. Figure 6d exemplifies the worst-case scenario, where the sorting of edges makes it so that there is no continuous *runway*. Therefore, the quality metric is not influenced by the number of nodes but number of edges. With an increasing density, chances are higher of concealing patterns or missing important connections. Although this simple heuristic works well, it does not distinguish between incomplete or disconnected *runways*.



	(a)	(b)	(c)	(d)
#runways:	1	1	2	0
completeness:	1	$(5/6)^2$	$(3/6)^2 + (3/6)^2$	n/a
quality:	1	0.69	0.5	0

Fig. 6: *Runways* are highlighted in ■ in the above pictures with additional statistics about the respective quality metric.

3.2 The Staircase

Staircases are a specific kind of *runways* having a more coherent visual structure additionally incorporating the Gestalt Law of *continuity*. They relate to *star* patterns in MSVs or NLDs and become apparent when the connections of a single node are ordered according to length. If multiple *staircases* are next to each other, the *sawtooth* pattern [39] emerges.

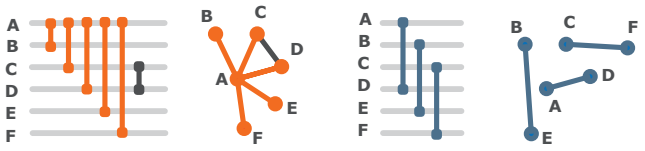


Fig. 7: The figure shows a comparison between a *staircase* and an *escalator* anti-pattern, highlighting the fact that the *escalator* can resemble a *staircase* structure. Still, it might be highly misleading, as it could actually lead to the belief that the involved elements do have a relationship, while in reality, they might not. In the image on the right, indeed, the involved elements are disconnected components.

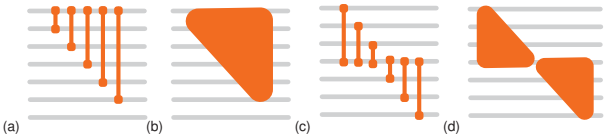


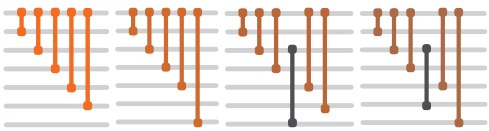
Fig. 8: *Staircases* can appear as triangles in BF. Note that a *staircase* can also cross its baseline, as seen in 8c and 8d. *Staircases* can also appear having different stepsizes, as seen in Figure 9.

To evaluate the quality of a *staircase* pattern, we take two different characteristics into account (i.e., *completeness* and *smoothness*). The *completeness* corresponds to the variance and describes how many connections of a single node are covered in a single *staircase*. The *smoothness* describes height differences between individual steps, thus influencing the perception of the Gestalt Law of *continuity*.

$$q_{staircases} = \sum^{\text{staircases}} \left(\left(\frac{\#steps+1}{degree} \right)^2 \times \left(1 - \frac{\sum^{\#steps} (d - min(d_{stair}))}{(\#steps) \times (max(d_{stair}) - min(d_{staircases}))} \right) \right) \quad (2)$$

The first part of Equation (2) covers the *completeness* of the *staircase* pattern. The *degree* corresponds to the number of connections of the node being in focus, and $\#steps$ corresponds to the number of imaginary horizontal lines connecting the vertical edges. This part of the formula will become optimal if all connections of a single node are part of the same *staircase*. The *completeness* is negatively influenced by the density of the graph since chances are higher that connections of a single node are split due to intruder edges.

The second part describes the *smoothness* of the staircase. The variable d measures the height of a single step. It describes the vertical distance between two neighboring edges contained in a *staircase* pattern. If edges next to each other belong to neighboring nodes, the value of $d = 1$ and is, therefore, optimal. Therefore, a bigger graph (i.e., having more nodes) is supposed to have a negative influence on this quality metric since chances are higher that neighboring edges do not belong to neighboring nodes. The sum of all step sizes is normalized. To avoid dividing by zero, we replace the entire term with 1, if $max(d_{stair}) == min(d_{stair})$. A comparison of configurations of different staircase qualities can be found in Figure 9. Like with the *runway* quality measure, we cannot distinguish whether a lower quality reflects an incomplete *staircase* or not optimal step sizes.



#staircases:	1	1	2	2
[min(d), max(d)]	[1,1]	[1,2]	[1,1]	[1,2]
smoothness:	1	0.75	1, 1	1, 0.5
completeness:	1	1	$(3/5)^2$, $(2/5)^2$	$(3/5)^2$, $(2/5)^2$
quality:	1	0.75	0.52	0.39

Fig. 9: **Quality assessment:** While these 4 pictures do not represent the same network (the connections between nodes change), they serve to show which configuration would be preferable over the others, going left to right in terms of overall quality.

Although Equation (2) provides a good generalization of the quality, it could make sense to restrict the maximum value of variable d in certain scenarios. Visualizing BF with fewer nodes introduces more space between the horizontal lines. Therefore, visually perceiving *staircases* with a high value for d is getting more difficult since the Gestalt Law of *proximity* might be violated. In contrast, the same value for d might be suitable for a bigger network with more nodes since the distance between the horizontal lines becomes smaller. Since the choice depends on many parameters (i.e., number of nodes, available screen space, thickness and distance between lines), we did not introduce a maximum cap for d and leave this for future research.

3.3 The Escalator

While analyzing *staircases*, we stumbled upon an artifact. The *escalator* looks like a combination of a *staircase* and a *band*; however, it represents an anti-pattern like the *bandwidth* or *noise* pattern in AMs. Its coherent appearance, which looks like a *parallelogram*, pretends to reveal a topological structure, although none exists.

Escalators emerge when neighboring edges of identical length are shifted in one position on the vertical axis. However, unlike *paths*, edges in *escalators* do not connect similar nodes. To minimize the presence of *escalators*, connections of a single node should be next to each other, similar to *runways*. Additionally, nodes should be ordered to avoid edges of identical length like in *staircases* (Figure 10).

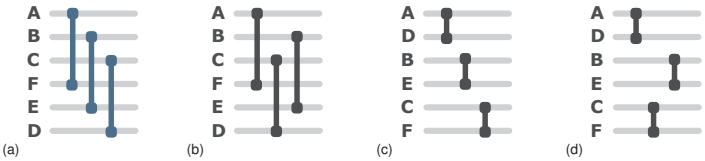


Fig. 10: We should look out for the *escalator* (10a) anti-pattern, because although it vaguely resembles a *staircase*, the nodes contained in it have no relation. Carefully sorting edges (10b), nodes (10c), or a combination of both (10d) can help reduce the appearance of the anti-pattern and reveal the absence of a relationship between the involved nodes.

3.4 The Podium

While experimenting with reordering techniques on how to avoid *escalators*, we created another anti-pattern, the *podium*. It looks like an isosceles triangle, similar to a rotated *staircase*; however, it does not convey any structural information.

The *podium* appears when pairs of unrelated nodes enclose each other on the vertical axis, and their corresponding edges are neighbors, therefore growing or shrinking in size. To minimize the presence of *podiums*, pairs of connected nodes should be positioned far from each other at varying distances. Also, unrelated edges should not be close. As we already said, *escalators* and *podiums* describe the same underlying data just with a different arrangement of nodes and edges (Figure 11).

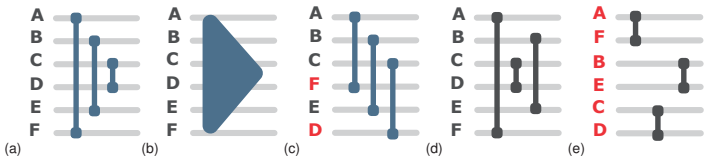


Fig. 11: The *podium* anti-pattern (11a) can look like a relevant motif, but it is highly misleading: its nodes have nothing in common, and if we switch the order of the nodes we can easily reveal that this is an *escalator* (11c). Changing the order of edges appropriately (11d) can reduce the appearance of the anti-pattern, and a sensible combination of both nodes and edges (11e) can reveal that these nodes are indeed unrelated.

3.5 Other shapes

Given the four patterns that we have investigated previously, one might also be curious about other possible representations of common patterns that appear in NLDs or AMs, (previously discussed in Section 2.1).

The Path: It represents the same topological structure as the *path* in NLDs and *bands* in AMs. Although the *path* looks similar to the *escalator*, they must not be confused. In contrast to the *escalator*, two neighboring edges in the *path* pattern share a similar node. That is why the starting and ending nodes are connected via different hops. The characteristic shape of a *path* looks like a thin *parallelogram* (Figure 12).

To reveal *paths*, the corresponding edges must be next to each other and of identical length. Therefore, nodes have to be neighbors, with the starting node being at the top and the target node at the bottom.

Fig. 12: How to draw a *path* in BF? Both a closed and open path could be highlighted through bands, ordering first through top node index, then bottom node index.

The Clique: Representing *cliques* or *communities* in BF is not a trivial task. Different recognizable structures emerge depending on the order of edges and nodes. While the two options presented in Figure 13 can look somewhat recognizable when presented in this simple context, it would be very easy to break the structure when the graph contains other elements in addition to the *clique* — e.g., it would be enough for a single edge that does not belong to the clique to be in the middle of the pattern to damage its recognizability. In addition, these structures would make it very easy for almost-cliques (cliques with one or a few missing edges) to be mistaken for full cliques, making it a dangerous structure to display. Additionally, it is unknown which structure is best perceived by humans in this context since systematic experiments about pattern recognition in BF are missing.

Fig. 13: How to draw a *clique* in BF? The image on the left represents the appearance of the structure in NLDs. The middle image is an attempt at representing a *clique* as a series of *staircases* of decreasing size - obtained through sorting the edges by top node index, then bottom node index. The rightmost represents it through *bands*, sorting the edges by shortest length. Any ordering of the node rows in a clique is irrelevant.

The Connector Similar to the *clique*, there are different visual structures possible to represent the *connector* pattern. If the *connector* consists of multiple nodes, two triangle shapes similar to the *staircase* pattern might emerge (Figure 14). The question of arranging these triangles cannot be easily answered since empirical evidence is missing on which overall shape to prefer.

Fig. 14: How to draw a *connector* motif in BF? The picture above shows two options to interlock the *staircases* that would be formed by the two nodes at the two ends of the connector.

4 Sorting Strategies in BioFabric

Although closely related to MSVs, BF visualizations are different since nodes and edges can be arranged individually, whereas MSVs rely on their temporal structure for positioning edges. This results in a complex challenge to tackle since both factors impact the quantity, quality, and types of patterns in the visualization. Unfortunately, node and edge arrangements have received little research attention; therefore, we cannot rely on previous results.

We started our endeavor by first looking into classical node ordering algorithms solving graph drawing problems (i.e., linear layouts like book embeddings). Indeed, we can see an *escalator* as a crossing in an arc layout, as seen in Figure 15.

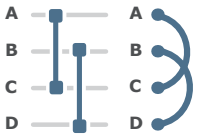


Fig. 15: An *escalator* would produce a crossing if the same ordering of nodes was represented on an arc diagram. We can theorize that crossing reduction strategies on arc diagrams would help us mitigate the appearance of the *escalator* anti-pattern.

Next, we considered AM reordering strategies since, in BF, we can also influence the layout by changing the order of elements over two axes. However, both approaches (NLDs and AMs reordering) have their own quirks that do not fully align with BF. For example, we can change rows and columns in AMs separately, consequently changing the node order. In BF, the tabular structure is slightly different since columns represent edges.

Other methods in addition to the ones we experiment with in this paper, are also possible. For instance, we believe that optimizing for *staircases* would represent an interesting problem that can be solved optimally through an Integer Linear programming formulation. However, we also believe that doing so would require a much longer discussion than what can be done in the limited space available in this paper; thus, we deem that as future work.

4.1 Node sorting methods

The ordering of nodes is going to greatly influence the outcome of the visualization [20,49] and, in combination with the edge ordering, will determine what patterns appear in the final visualization. Two properties in BF are uniquely influenced by the node order and not by the edge layout:

- The node layout determines edge length. No edge ordering can change edge length.
- The ordering of the nodes is going to determine the appearance of what would be "edge crossings" if the graph is seen as an arc diagram. However, the importance of these crossings is debatable.

Our choice of sorting algorithms is determined from previous literature on 1-dimensional NLDs, AMs, and MSVs reordering algorithms, which have been previously discussed in Section 2. In our selection, we considered six different approaches for revealing patterns. While some of these methods work more towards optimizing an individual single *staircase* (such as sorting nodes by degree), other strategies reduce the overall edge length (and produce more but smaller *staircases*).

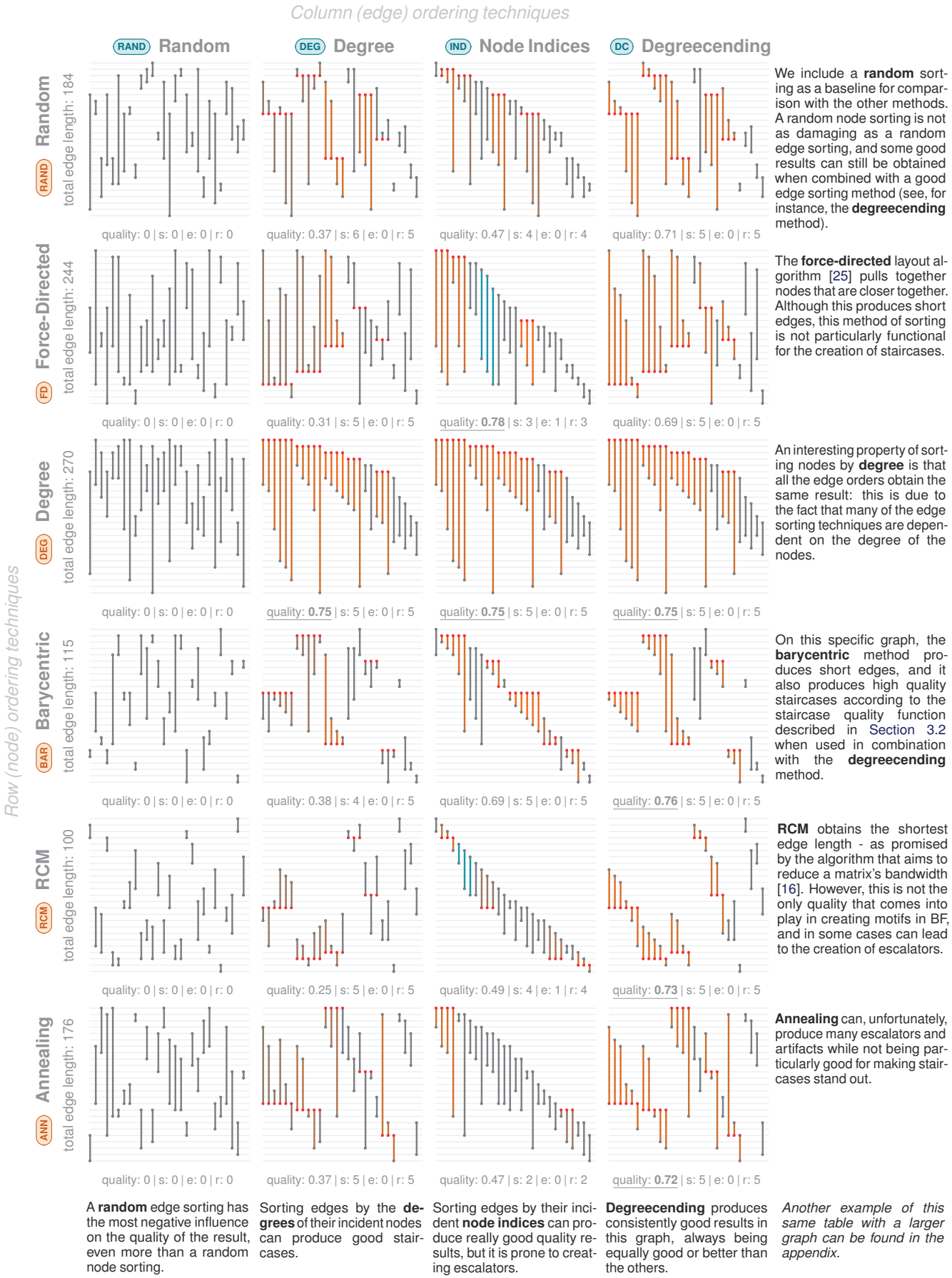


Fig. 16: The above visualizations are all comparisons of the effects of visualizing different nodes and edge sorting methods on the same graph, `grafo2700.25` from Rome-Lib [3], that contains 25 nodes and 28 edges. The color ■ indicates the presence of *staircases*, with the intensity of the color describing the staircase quality, while ■ highlights runways, and ■ escalators. In the captions of the images, `s` indicates the number of stairways, `e` the number of escalators, and `r` the number of runways. Underlined quality values are those above the 3rd quartile (75%) of the score obtained with other methods. While an example using an individual graph can't speak for general cases, it is still useful to showcase and discuss how different sorting methods affect the final result. The best combination for the above graph in terms of overall quality is BAR with DC .

RAND **Alphabetical / Random:** This ordering is meant as a baseline for testing the other configurations against a random sorting. Alphabetical would sort the nodes according to their *name* or *id* (which is useful in certain case studies), while RAND shuffles the nodes randomly using a seed.

DEG **Degree:** The nodes are sorted according to their degree, with the most connected node being placed at the top of the grid. This ordering helps to identify the most connected nodes quickly, and when combined with sorting edges by length or degree, it aims to produce the biggest staircase pattern for the node with the highest degree.

RCM **Reverse Cuthill-McKee:** The Cuthill-McKee algorithm [16] is a matrix reordering method aimed at reducing the bandwidth in AMs. It starts from a chosen root node, then runs a Breadth First Search on the nodes of the graph, assigning a value to them based on the order in which they are visited, and finally, sorts the entries in the matrix according to said value. *Reverse* Cuthill-McKee [30] is an improvement over it, where the ordering of the nodes is reversed compared to the assigned value and claims to produce an even smaller bandwidth. In the context of BF, smaller bandwidth will result in shorter edges. One possible concern regarding the usage of this algorithm is that its results vastly depend on what node is chosen to act as the starting (root) node for the application of the algorithm.

BAR **Barycentric:** The barycentric method works by placing every node at its neighboring nodes' median (or mean) position and repeating the process in iterations until convergence or a maximum iteration number is reached — resulting in nodes closer to their neighbors, shortening the edge length. We used the dot algorithm in Graphviz [27].

FD **Force-Directed:** We implemented a version of the Fruchterman-Reingold [25] force-directed layout algorithm operating on a single dimension. The purpose of this is similar to the barycentric method, although the logic used to attain the objective is vastly different — using a simulation of attractive and repulsive forces to simulate the behavior of springs. Although this method is usually displayed on at least two dimensions, it is easy to adapt it to work in only one by constraining the forces to apply to one individual dimension.

ANN **Simulated Annealing:** A probabilistic method used for finding an ordering of an AM's rows and columns to reveal hidden structures within a graph. It involves starting with an initial random or heuristic-based ordering and then iteratively generating new orderings by swapping rows and columns. The quality of an ordering is assessed using a predefined objective function. New orderings can be accepted even if they are worse than the current one, with a probability that decreases over time through a process mimicking metal cooling, hence the term "annealing." This allows the algorithm to escape local optima in the early stages while gradually focusing on fine-tuning towards a global optimum as the "temperature" decreases. The process continues until certain criteria, like a maximum number of iterations or a minimum temperature, are met. The final result is an optimized matrix ordering highlighting patterns, such as *cliques*, based on the chosen objective function [10].

We note that although many of these algorithms aim to cluster very connected nodes together or to reduce edge length, it might not be that minimizing edge length is what creates the most *staircases* or highlights any other property of the network properly. Therefore, we do not consider edge length reduction as one of our goals.

4.2 Edge sorting methods

The ability to sort edges is the new challenge introduced in BF. Although node sorting methods can be inspired by AM reordering strategies and NLD layout algorithms, edge sorting methods are kind of a new thing. We determined four different strategies to sort the edges, with the idea of formulating a heuristic to produce *staircase*-like structures.

RAND **Random:** As for the node sorting methods, we use a random sorting as a baseline to test against the other sorting methods.

DEG **Degree:** Edges are sorted according to the degrees of their incident nodes. The edge incident to the node with the highest degree is placed first. If both edges have the same highest degree (or are incident to the same node), the sorting considers the degrees of the other two incident nodes.

IND **Node indices:** Edges are sorted according to the position of the nodes in the current node order, prioritizing the node placed the highest. If two edges share the same highest node — which, in this case, can only happen if they are incident to the same node — the sorting considers the index of the other two incident nodes.

DC **Degreecending:** Edges are sorted according to the degree of their incident nodes first, then by length. The additional sorting by length should benefit *staircase* patterns.

Many other methods could also be possible: for instance, the possibility of sorting edges exclusively by their *length* comes to mind. However, we have to remind ourselves that our goal is to highlight properties of the network, and sorting edges by length would not particularly aid in spotting particular features in the network.

5 Experiment

To the best of our knowledge, node and edge arrangements in BF have received little research attention. There is no guidance on which combination of reordering techniques is best for conveying patterns [19]. Even small changes in the layout might ruin or reveal (anti-) patterns, as exemplified in Figure 6 or Figure 11. Therefore, we used a benchmark data set to compare different combinations of layout algorithms. Our experiment is the first to shed more light on this topic.

Layout algorithms: We took the well-established reordering algorithms introduced in Section 4. Where possible, we relied on libraries we could use to implement the algorithms (such as Graphviz [29], NetworkX [34], Reorder.js [24] and OGDF [12]), as opposed to implementing the algorithms ourselves wherever we could not find a viable alternative. When combining node and edge orderings, we follow a rule always to sort nodes first and edges last. As a result, we came up with 24 different combinations (i.e., 6 node orderings, 4 edge orderings). Figure 16 shows the result of applying the different combinations of node and edge orderings on the same data.

Data: We used the graph drawing test suite *Rome Graphs* [4], which contains around 11500 networks with different characteristics. It is a well-known benchmark dataset for evaluating aesthetic criteria and layout algorithms for NLDs [13, 14, 22]. The number of entities spans from a minimum of 10 to a maximum of 100 nodes with a varying density between 1% and 20%. We defined density as $d = \frac{e}{n(n-1)}$ where n denotes the number of nodes, and e is the number of edges [53]. Further information about the data can be found online [21].

Analysis: We focused on identifying *staircases* and *runways* to evaluate the effectiveness of all layout combinations. As an evaluation criterion, we counted the *number* of patterns found and their respective *quality* (Equation (1) and Equation (2)). To generalize the results and since the *number* of patterns highly correlates with the number of edges ($r(11532) = .96$, $p < .001$), we grouped the data by number of connections into four equal-width bins. The graph data is not distributed equally across the bins ($[0,40] = 2730$, $(40,80] = 4560$, $(80,120] = 2879$, $(120,160] = 1365$).

5.1 Results

A general overview of the results can be found in Table 1 with more detailed information being available in Table 2.

Overall performance: With an increasing number of edges and, therefore, patterns, the quality of the respective patterns is dropping. This is true for *staircases* ($r(11532) = -.30$, $p < .05$), as well as for *runways* ($r(11532) = -.67$, $p < .001$). Additionally, the quality of *runways* is negatively affected by their number ($r(11532) = -.50$, $p < .001$); this is not true for *staircases*. In general, the overall quality for *staircases* ($M: 0.53$, $IQR: 0.16$) is lower than for *runways* ($M: 0.91$, $IQR: 0.04$).

Edge orderings: Independent of the node order, RAND reveals nearly no *staircases* ($M: 0$, $IQR: 0$) nor *runways* ($M: 0$, $IQR: 0$). In contrast, DC shows the most *staircases* ($M: 12$, $IQR: 9.25$) with the highest quality ($M: 0.55$, $IQR: 0.06$), for all bins combined. DEG and IND perform similarly in detecting *staircases* ($M: 9.5$, $IQR: 8.75$ and $M: 9$, $IQR: 10.5$, respectively) with IND having the higher quality ($M: 0.54$, $IQR: 0.03$) compared to DEG ($M: 0.34$, $IQR: 0.083$). For finding *runways* in the data, DEG and DC perform equally well ($M: 12$, $IQR: 9.5$ and $M: 12$, $IQR: 9.25$) with an overall high quality ($M: 0.92$,

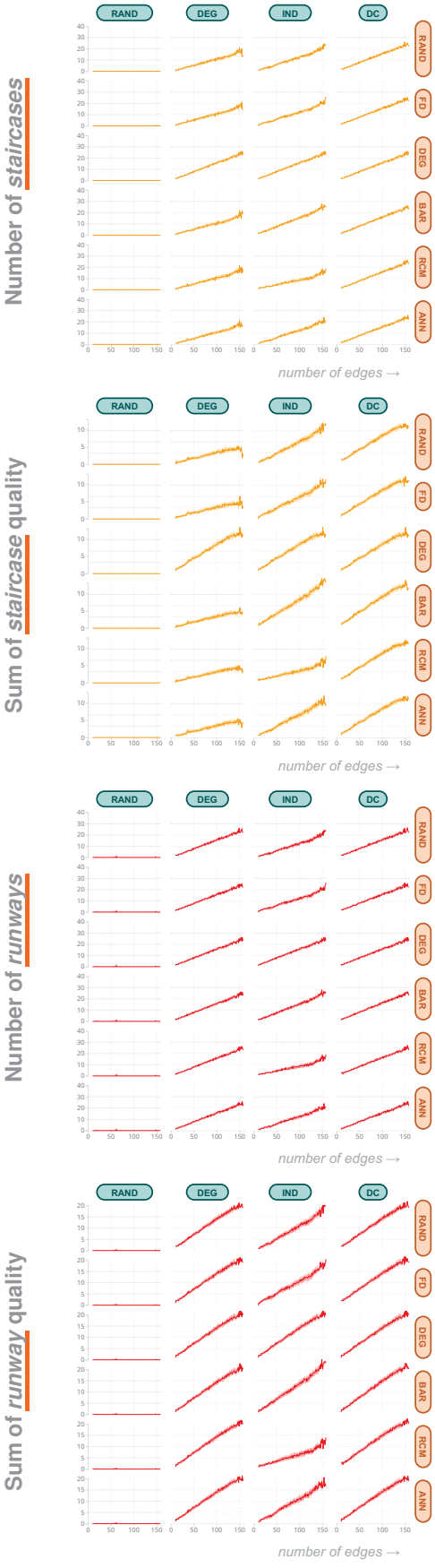


Table 1: These are the aggregate results when using all the combinations of node sorting methods and edge sorting methods over the entire corpus of Rome-Lib. The graphs are grouped by the number of edges growing on the x-axis. The line in the middle represents the median value, while the less saturated area represents quartiles.

IQR : 0.03 and M : 0.93, IQR : 0.05). IND finds fewer patterns (M : 9, IQR : 10.5) and with lower quality (M : 0.89, IQR : 0.03).

Node orderings: In contrast to the edge order, RAND as a node layout is not the worst for revealing *staircases* (M : 6.5, IQR : 11) or *runways* (M : 8, IQR : 13.5). RCM is performing even worse for *staircases* (M : 5.5, IQR : 8.75) and *runways* (M : 6.5, IQR : 10.75). The best-performing layout algorithm is DEG with the most patterns found (*staircases*: M : 9, IQR : 12.75; *runways*: M : 9, IQR : 12.75) and the highest quality (*staircases*: M : 0.54, IQR : 0.1; *runways*: M : 0.91, IQR : 0.1).

Combination: When looking at individual combinations of node and edge orderings, BAR together with DC shows the best performance for revealing *staircases* (number: M : 12.5, IQR : 9.5; quality: M : 0.58, IQR : 0.04). For detecting *runways*, RCM in combination with DC is best (number: M : 12.5, IQR : 9.5; quality: M : 0.93, IQR : 0.04). As already mentioned, RAND as edge layout should be avoided.

5.2 Discussion & Design Considerations

As can be seen from the results, the order of nodes and edges has a huge influence on revealing patterns in BF. An outlook on the scalability of our quality metrics regarding the number of nodes can be found in the appendix. We conducted an additional small pilot experiment with two much larger graphs comprising 4941 and 1174 nodes. The results are in line with our benchmark testing. Therefore, we suggest the following design considerations:

More edges result in more patterns with less quality: Of course, the underlying data has a huge influence on detecting patterns. Against intuition, the *density* of a network is less informative for judging layout strategies in revealing patterns. Two networks can have the same *density* but a different *number* of edges. With more edges, more substructures might be in the graph, and, therefore, more patterns are visible. However, with an increasing *number* of edges, the chances of concealing patterns by having an intruder edge are higher, thus lowering the overall quality (i.e., see Figure 9).

***Staircases* are more complex patterns than *runways*:** Although *runways* also depend on the edge ordering, the layout must not be as strict as for *staircases*. Therefore, their overall quality is higher. Since both patterns represent the same underlying topology, what is the additional value of *staircases*? It could be that the smoother shape of *staircases* is considered more salient, supporting the visual detection of patterns in BF. However, to find an answer to this question, a systematic, controlled user study must be conducted to investigate which pattern is best perceived and whether *runways* might be overlooked due to their irregular shape.

The edge order RAND provides the worst results: Even in combination with different node arrangements, nearly no patterns emerge. It comes as little surprise since neighboring edges should contribute to the same topological structure, which barely happens in a random arrangement. In contrast, although not optimal, RAND as a node arrangement can still show some patterns, especially with a suitable edge order like IND or DC . We expect this result to be also true for revealing different patterns like *paths* or *cliques*.

The combination of BAR and DC is best for revealing *staircases*: Moving well-connected nodes to their cluster center with related vertices being close enforces a compact representation with short neighboring edge lengths. Sorting the edges by length will reveal *staircases*. This combination of ordering techniques reveals, on average, 12.5 patterns with a median quality of 0.58, resulting in an overall quality of 7.25. The second-best combination scores only 6.6 in total. However, this ordering approach may not be as effective for *paths*, where the emphasis is not on individual nodes but rather on edges sharing a common target or source node.

The combination of RCM and DC is best for revealing *runways*: Results indicate that RCM with DC reveals the same number of *runways* as other combinations (M : 12.5), however, with a higher quality (M : 0.93). The overall quality (11.63) is a little better than the result of the direct successor (11.38). Since RCM reduces the bandwidth, edge lengths are minimized, positively influencing vertices' closeness. Connected nodes are positioned next to each other, enforcing *runway* patterns closely reflecting the node degree.

Node sorting ↓	Pattern	Edge sorting → RAND Random #Edges [0, 40]	RAND Random #Edges (40, 80]	RAND Random #Edges (80, 120]	RAND Random #Edges (120, 160]	DEG Degree #Edges [0, 40]	DEG Degree #Edges (40, 80]	DEG Degree #Edges (80, 120]	DEG Degree #Edges (120, 160]	IND Node Indices #Edges [0, 40]	IND Node Indices #Edges (40, 80]	IND Node Indices #Edges (80, 120]	IND Node Indices #Edges (120, 160]	DC Degreecending #Edges [0, 40]	DC Degreecending #Edges (40, 80]	DC Degreecending #Edges (80, 120]	DC Degreecending #Edges (120, 160]
RAND Random	#staircases	0	0	0	0	2	6	10	14	2	7	12	17	**3**	**[9]**	**15**	**20**
	staircase quality	0.24	0.24	0.17	0.17	0.4	0.34	0.32	0.3	0.57	0.54	**0.53**	**0.53**	**0.61**	**0.55**	**0.53**	0.52
	#runways	0	0	0	0	3	**[9]**	**15**	**20**	2	7	12	17	**[4]**	**[9]**	**15**	**20**
	runway quality	0.6	0.6	0.5	0.5	0.97	0.92	**0.91**	**0.89**	0.93	0.89	0.89	0.88	**[1]**	**0.93**	**0.91**	**0.89**
FD Force-Directed	#staircases	0	0	0	0	2	6	10	14	**3**	7	12	16	**3**	**[9]**	**15**	**20**
	staircase quality	0.21	0.19	0.12	0.12	0.4	0.34	0.32	0.3	0.57	0.54	0.53	**0.52**	**0.61**	**0.55**	**0.54**	**0.52**
	#runways	0	0	0	0	3	**[9]**	**15**	**20**	3	7	12	16	**[4]**	**[9]**	**15**	**20**
	runway quality	0.6	0.6	0.5	0.5	0.97	**0.93**	**0.91**	**0.89**	0.92	0.89	0.89	0.88	**[1]**	**0.93**	**0.91**	**0.89**
DEG Degree	#staircases	0	0	0	0	**3**	**[9]**	**15**	**20**	**3**	**[9]**	**15**	**20**	**3**	**[9]**	**15**	**20**
	staircase quality	0.24	0.24	0.17	0.17	**0.62**	**0.56**	**0.54**	**0.53**	**0.62**	**0.56**	**0.54**	**0.53**	**0.62**	**0.56**	**0.54**	**0.53**
	#runways	0	0	0	0	**3**	**[9]**	**15**	**20**	**3**	**[9]**	**15**	**20**	**3**	**[9]**	**15**	**20**
	runway quality	0.6	0.6	0.5	0.5	**[1]**	**0.93**	**0.91**	**[0.9]**	**[1]**	**0.93**	**0.91**	**[0.9]**	**[1]**	**0.93**	**0.91**	**[0.9]**
BAR Barycentric	#staircases	0	0	0	0	2	6	10	14	3	**[9]**	15	20	**[4]**	**[9]**	**[16]**	**[21]**
	staircase quality	0.24	0.17	0.17	0.23	0.41	0.35	0.33	0.31	0.63	0.56	0.54	0.54	**[0.65]**	**[0.59]**	**[0.57]**	**[0.56]**
	#runways	0	0	0	0	3	**[9]**	15	20	3	**[9]**	15	20	**[4]**	**[9]**	**[16]**	**[21]**
	runway quality	0.6	0.6	0.5	0.5	**[1]**	**0.93**	**0.91**	**[0.9]**	0.97	0.91	0.89	0.89	**[1]**	**0.93**	**0.91**	**[0.9]**
RCM RCM	#staircases	0	0	0	0	2	6	10	14	2	5	8	11	**3**	**[9]**	**15**	**20**
	staircase quality	0.24	0.24	0.14	0.17	0.38	0.32	0.31	0.29	0.54	0.44	0.41	0.4	**0.62**	**0.56**	**0.54**	**0.53**
	#runways	0	0	0	0	3	**[9]**	**[16]**	20	2	5	8	11	**[4]**	**[9]**	**[16]**	**[21]**
	runway quality	0.6	0.6	0.5	0.5	**[1]**	0.93	0.91	**[0.9]**	0.89	0.8	0.77	0.75	**[1]**	**[0.94]**	**[0.92]**	**[0.9]**
ANN Annealing	#staircases	0	0	0	0	2	6	10	14	2	7	12	17	**3**	**[9]**	**15**	**20**
	staircase quality	0.17	0.24	0.17	0.17	0.4	0.34	0.32	0.31	0.57	0.54	0.53	**0.53**	**0.61**	**0.56**	**0.54**	0.52
	#runways	0	0	0	0	3	**[9]**	**15**	**20**	2	7	12	17	**[4]**	**[9]**	**15**	**20**
	runway quality	0.6	0.6	0.5	0.5	**[1]**	0.92	**0.91**	**0.89**	0.92	0.89	0.89	0.88	**[1]**	**0.93**	**0.91**	**0.89**

Table 2: Median values obtained for #*staircases*, overall *staircase* quality, #*runways*, and overall *runway* quality for each combination of edge sorting methods and node sorting methods. The results are further split into graphs of different sizes based on the number of edges they contain. A bold **[number]** encased in square brackets indicates the best result overall for a given graph size across all combinations of node and edge sorting methods. A bold **number** with no square brackets indicates the best result given a specific node sorting method. We can see how the majority of best results are concentrated in the DC edge sorting method when combined with BAR or RCM .

6 Discussion & Limitation

Due to the sheer size of the problem at hand, we do not consider this paper to be an exhaustive essay on all the issues and possibilities associated with sorting nodes and edges in BF – rather, we would like to think of the open question of how to handle the problem as an exciting opportunity for a lot of new research and questions to answer.

Graph properties: The paper exclusively deals with graphs that allow one edge between two nodes. Allowing more than one edge between nodes would create the possibility for new shapes — for instance, squares. The exploration of how to obtain these other shapes and what graph structures map to them is a whole new can of worms. Additionally, we did not consider dynamic networks with temporal changes in the underlying topology. Patterns and ordering strategies will be different as can be seen in MSVs.

Order of sorting: We only consider sorting strategies where the nodes are sorted first, and the edges second — and where many of the edge sorting strategies are dependent on the order of nodes.

Optimal step size: As discussed in Section 3.2, the parameter d might be limited to a certain value to improve pattern recognition. In our experiment, we did not decide on a fixed maximum cap but kept it volatile. Depending on the underlying data, specifically the number of nodes, we expect a certain maximal cap as beneficial for perceiving *staircases*.

Quality metrics: The experiment was limited to assessing the quality of *runways* and *staircases* without a deeper investigation into whether our heuristics reflect users' intuition. A user study would be necessary to determine if these metrics accurately reflect user perceptions and preferences like the *smoothness* of the *staircases*.

Further, we did not consider other potential patterns such as *paths*. We expect results to be different when focusing on other patterns. However, we can already tell that there is not a single optimal reordering strategy revealing all possible patterns. As Figure 13 shows, we can either focus on showing *paths* or *staircases* since the patterns are mutually exclusive.

7 Conclusion & Future Work

In this paper, we identified patterns and anti-patterns in BF, such as *staircases*, *runways*, and *escalators*, and proposed quality metrics to evaluate different node and edge arrangements automatically. We conducted an experiment with a well-known benchmark data set, including approximately 11500 graphs with varying characteristics. Our findings suggest that the edge order has a stronger impact on revealing patterns than node layout. Through our experiment with well-established reordering techniques, we came to the conclusion that *staircase* patterns emerge best using BAR paired with DC and *runways* with RCM combined with DC . Throughout our studies, we distilled three promising future research directions.

User experiment: A comprehensive user study should be conducted to investigate which patterns are perceivable in BF visualizations. By collecting qualitative and quantitative feedback, we can gain valuable insights into the effectiveness of BF in conveying different patterns like *cliques* or *paths*. Understanding user perceptions will help to refine ordering techniques to better align with user expectations and tasks.

Motif simplification: Since some topological structures are more complex than others, motif simplification might help to improve pattern recognition. By reducing the complexity of patterns while preserving their essential characteristics, we can enhance the saliency of patterns and facilitate their identification. Investigating alternative simplification strategies and evaluating their impact on pattern recognition with real users will help to distill design considerations for representing stylized topological structures in BF.

Edge ordering As our results indicate, edge arrangements have a huge impact on revealing patterns. Investigating different edge ordering techniques tailored to different pattern types and evaluating their efficacy in enhancing pattern perception will be beneficial to better understanding topological structures with BF.

Acknowledgments

This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2117 – 422037984 and TRR 161 (Project A03) – Project-ID 251654672 and the Austrian Science Fund (FWF)[ESP 513-N].

References

[1] D. Auber, Y. Chiricota, F. Jourdan, and G. Melancon. Multiscale visualization of small world networks. In *IEEE Symposium on Information Visualization 2003 (IEEE Cat. No.03TH8714)*, pp. 75–81, 2003. doi: 10.1109/INFVIS.2003.1249011 1

[2] A. Barsky, T. Munzner, J. Gardy, and R. Kincaid. Cerebral: Visualizing multiple experimental conditions on a graph with biological context. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1253–1260, 2008. doi: 10.1109/TVCG.2008.117 1

[3] G. D. Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *Computational Geometry*, 7(5-6):303–325, Apr. 1997. doi: 10.1016/s0925-7721(96)00005-3 6

[4] G. D. Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *Computational Geometry*, 7(5-6):303–325, Apr. 1997. doi: 10.1016/s0925-7721(96)00005-3 7

[5] M. Behrisch, B. Bach, N. Henry Riche, T. Schreck, and J.-D. Fekete. Matrix reordering methods for table and network visualization. *Computer Graphics Forum*, 35(3):693–716, 2016. doi: 10.1111/cgf.12935 1, 2, 3

[6] M. Behrisch, B. Bach, M. Hund, M. Delz, L. Von Rüden, J.-D. Fekete, and T. Schreck. Magnostics: Image-based search of interesting matrix views for guided network exploration. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):31–40, 2017. doi: 10.1109/TVCG.2016.2598467 1, 3

[7] M. Behrisch, M. Blumenschein, N. W. Kim, L. Shao, M. El-Assady, J. Fuchs, D. Seebacher, A. Diehl, U. Brandes, H. Pfister, T. Schreck, D. Weiskopf, and D. A. Keim. Quality metrics for information visualization. *Computer Graphics Forum*, 37(3):625–662, 2018. doi: 10.1111/cgf.13446 2

[8] S. P. Borgatti and M. G. Everett. A graph-theoretic perspective on centrality. *Social Networks*, 28(4):466–484, 2006. doi: 10.1016/j.socnet.2005.11.005 2

[9] U. Brandes. *Force-Directed Graph Drawing*, pp. 1–6. Springer US, Boston, MA, 2008. doi: 10.1007/978-3-642-27848-8_648-1 2

[10] M. J. Brusco, H.-F. Köhn, and S. Stahl. Heuristic implementation of dynamic programming for matrix permutation problems in combinatorial data analysis. *Psychometrika*, 73(3):503–522, 2008. doi: 10.1007/s11336-007-9049-5 3, 7

[11] C.-H. Chen. Generalized association plots: Information visualization via iteratively generated correlation matrices. *Statistica Sinica*, 12(1):7–29, 2002. 3

[12] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, and P. Mutzel. The open graph drawing framework (ogdf). *Handbook of graph drawing and visualization*, 2011:543–569, 2013. 7

[13] M. Chimani, C. Gutwenger, P. Mutzel, and H.-M. Wong. Layer-free upward crossing minimization. *ACM J. Exp. Algorithmics*, 15, mar 2010. doi: 10.1145/1671970.1671975 7

[14] M. Chimani, P. Mutzel, and I. Bomze. A new approach to exact crossing minimization. In D. Halperin and K. Mehlhorn, eds., *Algorithms - ESA 2008*, pp. 284–296. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi: 10.1007/978-3-540-87744-8_24 7

[15] L. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1367–1372, 2004. doi: 10.1109/TPAMI.2004.75 2

[16] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference*, ACM '69, p. 157–172. Association for Computing Machinery, New York, NY, USA, 1969. doi: 10.1145/800195.805928 6, 7

[17] T. N. Dang, N. Pendar, and A. G. Forbes. Timearcs: Visualizing fluctuations in dynamic networks. *Computer Graphics Forum*, 35(3):61–69, 2016. doi: 10.1111/cgf.12882 3

[18] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *ACM Trans. Graph.*, 15(4):301–331, oct 1996. doi: 10.1145/234535.234538 2

[19] S. Di Bartolomeo, T. Crnovrsanin, D. Saffo, E. Puerta, C. Wilson, and C. Dunne. Evaluating graph layout algorithms: A systematic review of methods and best practices. *Computer Graphics Forum*, n/a(n/a):e15073, 2024. doi: 10.1111/cgf.15073 7

[20] S. Di Bartolomeo, A. Pister, P. Buono, C. Plaisant, C. Dunne, and J.-D. Fekete. Six methods for transforming layered hypergraphs to apply layered graph layout algorithms. *Computer Graphics Forum*, 41(3):259–270, 2022. doi: 10.1111/cgf.14538 5

[21] S. Di Bartolomeo, E. Puerta, C. Wilson, T. Crnovrsanin, and C. Dunne. A collection of benchmark datasets for evaluating graph layout algorithms. Under submission to Graph Drawing Posters, 2023. https://visdunneright.github.io/gd_benchmark_sets/, last accessed on 2024-03-26. 7

[22] S. di Bartolomeo, M. Riedewald, W. Gatterbauer, and C. Dunne. Stratisfimal layout: A modular optimization model for laying out layered node-link network visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):324–334, 2022. doi: 10.1109/TVCG.2021.3114756 7

[23] C. Dunne and B. Shneiderman. Motif simplification: improving network visualization readability with fan, connector, and clique glyphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, p. 3247–3256. Association for Computing Machinery, New York, NY, USA, 2013. doi: 10.1145/2470654.2466444 2

[24] J.-D. Fekete. Reorder.js: A JavaScript Library to Reorder Tables and Networks. IEEE VIS 2015, Oct. 2015. Poster. 7

[25] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. doi: 10.1002/spe.4380211102 2, 6, 7

[26] J. Fuchs, F. L. Dennig, M.-V. Heinle, D. A. Keim, and S. Di Bartolomeo. Exploring the design space of biofabric visualization for multivariate network analysis. *Computer Graphics Forum*, 43(3):e15079, 2024. doi: 10.1111/cgf.15079 1

[27] E. Gansner, E. Koutsofios, S. North, and K.-P. Vo. A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19(3):214–230, 1993. doi: 10.1109/32.221135 3, 7

[28] E. R. Gansner and Y. Koren. Improved circular layouts. In M. Kaufmann and D. Wagner, eds., *Graph Drawing*, pp. 386–398. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi: 10.1007/978-3-540-70904-6_37 2

[29] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software: Practice and Experience*, 30(11):1203–1233, 2000. doi: 10.1002/1097-024X(200009)30:11<1203::AID-SPE338>3.0.CO;2-N 7

[30] A. George and J. W. Liu. *Computer Solution of Large Sparse Positive Definite*. Prentice Hall Professional Technical Reference, 1981. doi: 10.1137/1026055 7

[31] J. A. George. *Computer Implementation of the Finite Element Method*. PhD thesis, Stannford University, Stanford, 1971. 3

[32] J. A. Grochow and M. Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *Proceedings of the 11th Annual International Conference on Research in Computational Molecular Biology*, RECOMB'07, p. 92–106. Springer-Verlag, Berlin, Heidelberg, 2007. doi: 10.1007/978-3-540-71681-5_7 2

[33] X. Guan, C. Wu, W. Yang, and J. Meng. A survey on book-embedding of planar graphs. *Frontiers of Mathematics in China*, 17(2):255–273, 2022. doi: 10.1007/s11464-022-1010-5 2

[34] A. Hagberg, P. J. Swart, and D. A. Schult. Exploring network structure, dynamics, and function using networkx. 1 2008. 7

[35] 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 2

[36] B. Kerr. Thread arcs: an email thread visualization. In *IEEE Symposium on Information Visualization 2003 (IEEE Cat. No.03TH8714)*, pp. 211–218, 2003. doi: 10.1109/INFVIS.2003.1249028 2

[37] C. D. Linhares, J. R. Ponciano, F. S. Pereira, L. E. Rocha, J. G. S. Paiva, and B. A. Travençolo. A scalable node ordering strategy based on community structure for enhanced temporal network visualization. *Computers & Graphics*, 84:185–198, 2019. doi: 10.1016/j.cag.2019.08.006 3

[38] C. D. G. Linhares, B. A. N. Travençolo, J. G. S. Paiva, and L. E. C. Rocha. Dynetvis: a system for visualization of dynamic networks. In *Proceedings of the Symposium on Applied Computing*, SAC '17, p. 187–194. Association for Computing Machinery, New York, NY, USA, 2017. doi: 10.1145/3019612.3019686 3

[39] W. J. Longabaugh. Combing the hairball with BioFabric: a new approach for visualization of large networks. *BMC Bioinformatics*, 13(1):275, Oct.

2012. doi: 10.1186/1471-2105-13-275 1, 3, 4

[40] E. Mäkinen and H. Siirtola. The barycenter heuristic and the reorderable matrix. *Informatica (Slovenia)*, 29(3):357–364, 2005. 3

[41] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002. doi: 10.1126/science.298.5594.824 2

[42] C. Nobre, M. Meyer, M. Streit, and A. Lex. The state of the art in visualizing multivariate networks. *Computer Graphics Forum*, 38(3):807–832, 2019. doi: 10.1111/cgf.13728 1

[43] C. Nobre, D. Wootton, L. Harrison, and A. Lex. Evaluating multivariate network visualization techniques using a validated design and crowdsourcing approach. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, p. 1–12. Association for Computing Machinery, New York, NY, USA, 2020. doi: 10.1145/3313831.3376381 1, 2

[44] C. Pich. *Applications of Multidimensional Scaling to Graph Drawing*. PhD thesis, Universität Konstanz, Konstanz, 2009. 3

[45] H. C. Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing*, 13(5):501–516, 2002. doi: 10.1006/jvlc.2002.0232 1

[46] D. Satsangi, K. Srivastava, and G. Srivastava. $$k$$-page crossing number minimization problem: An evaluation of heuristics and its solution using gesakp. *Memetic Computing*, 5(4):255–274, 2013. doi: 10.1007/s12293-013-0115-5 2

[47] J. M. Six and I. G. Tollis. A framework and algorithms for circular drawings of graphs. *Journal of Discrete Algorithms*, 4(1):25–50, 2006. doi: 10.1016/j.jda.2005.01.009 2

[48] H. Turner, T. Bailey, and W. Krzanowski. Improved biclustering of microarray data demonstrated through systematic performance tests. *Computational Statistics & Data Analysis*, 48(2):235–254, 2005. doi: 10.1016/j.csda.2004.02.003 3

[49] P. Valdivia, P. Buono, C. Plaisant, N. Dufournaud, and J.-D. Fekete. Analyzing dynamic hypergraphs with parallel aggregated ordered hypergraph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 27(1):1–13, 2021. doi: 10.1109/TVCG.2019.2933196 5

[50] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. Reordering massive sequence views: Enabling temporal and structural analysis of dynamic networks. In *2013 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 33–40, 2013. doi: 10.1109/PacificVis.2013.6596125 3

[51] C. Ware, H. Purchase, L. Colpoys, and M. McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002. doi: 10.1057/palgrave.ivs.9500013 1

[52] M. Wattenberg. Arc diagrams: visualizing structure in strings. In *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.*, pp. 110–116, 2002. doi: 10.1109/INFVIS.2002.1173155 2

[53] V. Yoghourdjian, D. Archambault, S. Diehl, T. Dwyer, K. Klein, H. C. Purchase, and H.-Y. Wu. Exploring the limits of complexity: A survey of empirical studies on graph visualisation. *Visual Informatics*, 2(4):264–282, 2018. doi: 10.1016/j.visinf.2018.12.006 7