Bundling-Aware Graph Drawing

Daniel Archambault
Newcastle University, UK

Giuseppe Liotta
University of Perugia, Italy

Martin Nöllenburg
TU Wien, Austria

Tommaso Piselli
University of Perugia, Italy

Alessandra Tappini
University of Perugia, Italy

Markus Wallinger
TU Munich, Germany

Abstract

Edge bundling algorithms significantly improve the visualization of dense graphs by reducing the clutter of many edges visible on screen by bundling them together. As such, bundling is often viewed as a post-processing step applied to a drawing, and the vast majority of edge bundling algorithms consider a graph and its drawing as input. Another way of thinking about edge bundling is to simultaneously optimize both the drawing and the bundling. In this paper, we investigate methods to simultaneously optimize a graph drawing and its bundling. We describe an algorithmic framework which consists of three main steps, namely *Filter*, *Draw*, and *Bundle*. We then propose two alternative implementations and experimentally compare them against the state-of-the-art approach and the simple idea of drawing and subsequently bundling the graph. The experiments confirm that bundled drawings created by our framework outperform previous approaches according to standard quality metrics for edge bundling.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Human-centered computing → Graph drawings

Keywords and phrases Edge Bundling, Experimental Comparison, Graph Sparsification

Digital Object Identifier 10.4230/LIPIcs.GD.2024.15

Supplementary Material *Software*: `https://doi.org/10.17605/OSF.IO/G4XQW` [36]

Funding *Giuseppe Liotta*: MUR PON Proj. ARS01_00540; MUR PRIN Proj. 2022TS4Y3N; MUR PRIN Proj. 2022ME9Z78.
Martin Nöllenburg: Vienna Science and Technology Fund (WWTF) [10.47379/ICT19035].
Tommaso Piselli: MUR PON Proj. ARS01_00540; MUR PRIN Proj. 2022TS4Y3N; MUR PRIN Proj. 2022ME9Z78.
Alessandra Tappini: MUR PON Proj. ARS01_00540; MUR PRIN Proj. 2022TS4Y3N; MUR PRIN Proj. 2022ME9Z78.
Markus Wallinger: Vienna Science and Technology Fund (WWTF) [10.47379/ICT19035].

1 Introduction

The majority of bundling algorithms consider both the graph and its drawing as input [23]. With the notable exception of approaches that are inspired by confluent drawings [2, 41], the coordinates of the nodes are vital for computing a bundled drawing of high quality. However, often we would like to produce a graph layout that simultaneously optimizes the drawing and

the bundling. For some graphs, edge bundling will not be necessary: if we have a sufficiently good drawing of a graph, there is no need to bundle it. In the case of dense graphs, it will be very difficult to find a drawing of the graph that does not require some or many of the edges to be bundled to resolve its hairball-like appearance. In such cases, it may be advantageous for us to consider bundling alongside drawing.

One way to optimize the bundling alongside the drawing is to select edges that are likely to be bundled, remove them from the graph, and draw the skeleton of the edges that will be bundled against. Edge-Path bundling (`EPB`) algorithms [37, 38] have such a skeleton with edges bundled against paths. In this case, the smaller graph that would be computed for layout would be the edges that participate in the paths used for bundling. The confluent drawing inspired approaches of Bach et al. [2] and Zheng et al. [41] perform aggregation via a power set decomposition and create a hierarchy of coarse graphs that are used to draw and bundle the graph in a confluent way. This aggregation approach will help focus graph layouts on edges that can support bundles. However, confluent drawings have a lower degree of bundling when compared to `EPB`, and it may be interesting to investigate approaches with a greater degree of bundling.

In this paper, we contribute a framework which offers a greater degree of bundling than confluent-like approaches while optimizing bundling alongside drawing. Two instantiations of this framework are explored. Our `Filter-Draw-Bundle` framework selects nodes and edges that form the scaffold of the bundling, or the set of edges in the graph that will be bundled against. Edges that are more likely to be bundled are filtered from the graph. As Edge-Path bundling has a very strong connection to t-spanners [1, 38], we show how various levels of t-spanners can be used for drawing with the remaining edges bundled onto the spanner.

2 Related Work

Edge bundling algorithms have been studied for nearly 20 years now [16] with many approaches [23] created to simplify edge clutter of densely connected networks. Most edge bundling techniques [4, 9, 10, 17, 19, 20, 22, 24, 27, 29–32, 35, 39] consider a drawing and a graph as input and bundle edges together based on similar properties such as orientation, co-location in space, and other similar edge properties. Recent Edge-Path bundling techniques [37, 38] instead bundle long edges with paths. The main advantage for Edge-Path bundling is that it does not create the illusion of false connections between disconnected edges as the subtended edge always has a corresponding path. In all of the above cases, a drawing of the graph is taken as input and is used for bundling. This has been successfully applied in practice, but considering the layout in combination with bundling has not been investigated before. Furthermore, our results show that creating a layout more tailored for bundling has better performance regarding certain quality metrics.

Confluent drawings [6, 7, 14] do not consider a drawing beforehand. Rather, the graph is converted into a planar graph by contracting down bicliques, drawing the high level planar graph, and reintroducing the bicliques into the graph so that they can be bundled without ambiguity. Less restrictive versions of these drawings [2, 33, 41] do not require the strict planar condition and generally compute a decomposition of the graph into components. The decomposition of the graph is drawn and then the edges between components are bundled. While heuristics for computing confluent drawings introduce less ambiguity compared to our proposed framework, they are much more constrained in what can be bundled, leading to less overall bundling.

In the work presented here, we focus on a new way of optimizing a bundling while drawing the graph simultaneously. In our case, we filter the graph in a way that takes away edges in an Edge-Path bundling that are likely to be bundled, providing a low-stress layout of the paths to be bundled against. In our experiment, we compare against the state-of-the-art methods that produce confluent-like drawings [2, 41] as well as against the straight-forward approach of first creating a low-stress drawing of the full input graph and then applying Edge-Path bundling on it.

3 The Filter-Draw-Bundle Framework

In this section, we introduce the `Filter-Draw-Bundle` (`FDB`) Framework. This framework is designed to optimize three well-studied quality metrics for bundled drawings, namely the ink ratio, distortion, and ambiguity. We start by formally defining these metrics, since we shall refer to them when describing the framework.

3.1 Bundling Quality Metrics

We consider three metrics that are in line with the ones described by Wallinger et al. [37, 38], namely *ink ratio*, *distortion*, and *ambiguity*. Smaller values in all three metrics are considered better. Let $G = (V, E)$ be the input graph, let $\mathcal{A}_B$ be a bundling algorithm and let $\mathcal{L}_B$ be the layout obtained by applying $\mathcal{A}_B$ to G. The metrics we consider are defined as follows.

Ink Ratio. Informally, the ink ratio is the proportion of pixels within the drawing's bounding box that are occupied by $\mathcal{L}_B$. It can quantify simplification through a measure of data-ink ratio [34]. More precisely, let $L_B \in \{0, \dots, 255\}^{m \times n}$ be an $m \times n$-greyscale bitmap image of $\mathcal{L}_B$, and let $0 \leq \delta < 255$ be a threshold grey value below which we consider a pixel occupied; in our implementation we consider $\delta = 254$, which means all non-white pixels are occupied. We scale the drawing $\mathcal{L}_B$ to a bitmap that has a width of 1.000 pixels when computing the ink ratio. For each pixel $L_B[i, j]$ of L_B, where $1 \leq i \leq m$ and $1 \leq j \leq n$, we set $p_B(i, j) = 1$ if $L_B[i, j] \leq \delta$, otherwise we set $p_B(i, j) = 0$. The ink ratio of L_B is then defined as:

$$\text{InkRatio} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} p_B(i, j)}{m \cdot n}. \tag{1}$$

We note that our definition of ink ratio is slightly different from Wallinger et al. [37, 38], who measured the reduction in occupied pixels compared to a straight-line input drawing. Since we do not consider any input drawing for the `FDB` framework, their definition is not applicable.

Distortion. The distortion of an edge is the ratio between its length in $\mathcal{L}_B$ and the Euclidean distance of its endpoints. Human-centred studies have shown that deviations from straight line distances make paths harder to read [18]. Let $\ell_B(uv)$ be the length of an edge uv in $\mathcal{L}_B$ and let $d(u, v)$ be the Euclidean distance between u and v. The distortion of $\mathcal{L}_B$ is the average distortion of its edges, namely:

$$\text{Distortion} = \frac{1}{|E|} \sum_{uv \in E} \frac{\ell_B(uv)}{d(u, v)}. \tag{2}$$

Ambiguity. The ambiguity of a bundled drawing measures how many false adjacencies could be derived erroneously from the drawing $\mathcal{L}_B$. Avoiding such false connections would be considered more faithful [26]. More precisely, the ambiguity is defined as the ratio between

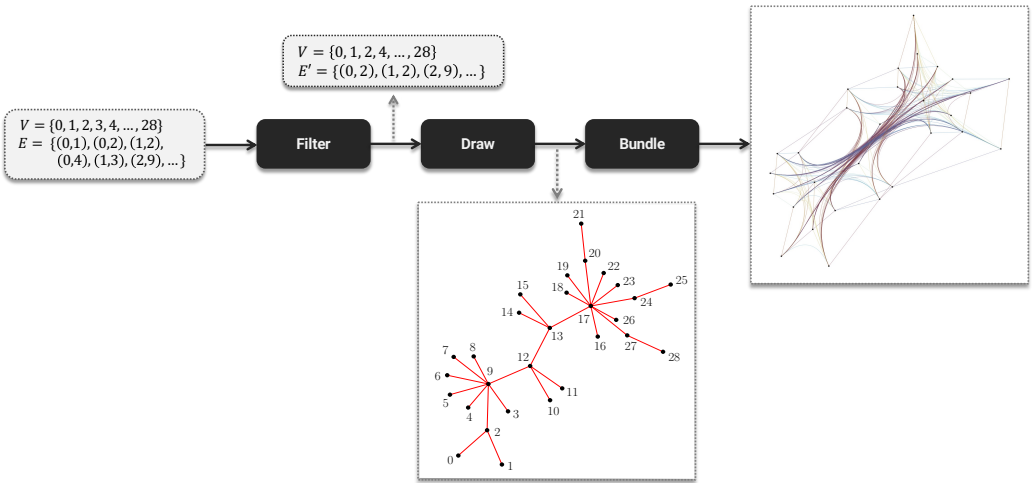

Figure 1 Illustration of our FDB framework. The `Filter` step takes as input an abstract graph $G = (V, E)$ and outputs a sparsification $G' = (V, E')$ with $E' \subseteq E$; the `Draw` step computes a drawing Γ of G'; the `Bundle` step adds to Γ the edges in E' and bundles them.

the number of perceivable false adjacencies and the total number of perceivable adjacencies. Here, an adjacency between two vertices u, v is a false adjacency if the topological distance between u and v is greater than a certain threshold. Given an edge $e = st \in E$, which is drawn as a curve in $\mathcal{L}_B$, the set $N_B(s, e)$ of reachable neighbors of s along e consists of all the vertices that are perceived as being connected to s. More precisely, a vertex v belongs to $N_B(s, e)$ if there is an edge $e' = uv \in E$ that is drawn in $\mathcal{L}_B$ as a curve that intersects e by forming a flat angle which is smaller than a certain threshold θ. The set $N_B(s, e)$ may contain both true and false neighbors of s: we denote as $N_B^{\mathsf{T}}(s, e)$ the set of the true neighbors and as $N_B^{\mathsf{F}}(s, e) = N_B(s, e) \setminus N_B^{\mathsf{T}}(s, e)$ the set of the false neighbors. We consider two vertices as true neighbors if the length of the shortest path between them in G (in hop distance) is smaller than a certain threshold δ. So if $\delta = 1$ (as in or implementation), $N_B^{\mathsf{T}}(s, e)$ contains only the direct neighbors of s, otherwise it contains all vertices reachable in hop distance at most δ. The ambiguity of $\mathcal{L}_B$ is then defined as:

$$\text{Ambiguity} = \frac{\sum_{s \in V} \sum_{e=st \in E} |N_B^{\mathsf{F}}(s, e)|}{\sum_{s \in V} \sum_{e=st \in E} |N_B(s, e)|}. \tag{3}$$

3.2 Framework Description

We are now ready to describe our algorithmic framework, which we call `Filter-Draw-Bundle`, or `FDB` for short. The algorithmic framework is illustrated in Figure 1 and summarized in Algorithm 1. As the name suggests, it consists of the following three steps:

- `Filter:` Let $G = (V, E)$ be the input graph. In the `Filter` step, a subgraph G' of G is computed, with $G' = (V, E')$ such that $V' = V$ and $E' \subseteq E$. In particular, E' is supposed to be the subset of those edges of G against which the edges of $E \setminus E'$ will be bundled in the third step (in our case to optimize for Edge-Path bundling). Graph G' is called the *skeleton* of G.
- `Draw:` A drawing Γ of the skeleton G' is computed. The quality metrics of the subsequent bundled graph drawing will strongly depend on the layout of the skeleton.

- `Bundle`: Γ is enriched with those edges of G that were filtered out in the first step of `FDB`. In the instantiations that follow, we shall adopt the Edge-Path bundling (`EPB`) approach to bundle against the edges of the skeleton G'.

Algorithm 1 The `FDB` Framework.

Input: Graph $G = (V, E)$.
Output: A bundled layout $\mathcal{L}_B$ of G.
`/* Filter Step */`
$w \leftarrow$ computeWeights(G)
$G' = (V, E') \leftarrow$ sparsifyGraph(G, w) `// G' is the skeleton of G`
`/* Draw Step */`
$\Gamma \leftarrow$ computeDrawing(G')
`/* Bundle Step */`
$\mathcal{L}_B \leftarrow$ edgePathBundling(G, Γ)

In contrast to other bundling approaches, the `FDB` framework does not receive the vertex coordinates of the final layout as part of the input, but it computes them during the `Draw` step. Hence, the `FDB` approach is not a post-processing procedure that improves the readability of a given drawing, but it takes into account the quality metrics of the target bundling during each step of its algorithmic pipeline, and in particular aims to find a drawing of a suitable skeleton graph that gives rise to a high-quality bundling.

It is also worth recalling that Bach et al. [2] and Zheng et al. [41] describe similar ideas. Namely, they first compute a hierarchical aggregation of the graph, then draw such an aggregation, and finally compute a confluent drawing. While the `Filter` step of the `FDB` pipeline computes a subgraph of G, the approach of [2, 41] suitably selects groups of nodes, aggregates them, and connects such groups with dummy edges. Once a layout of the hierarchical aggregation is computed, a confluent drawing of the graph is constructed by using these dummy edges as a guideline.

3.2.1 Instantiating the `FDB` Framework

While the flexibility of our algorithmic framework makes it possible to adopt different filtering, drawing, and bundling strategies, as a proof of concept we make specific choices for the three different steps which will be used in our experimental analysis. Namely, we describe two different instances of the `FDB` framework, which differ for the adopted strategy in the `Filter` step. In both cases, we consider filter and draw steps that optimize `EPB`. The framework can be used for other bundling algorithms, but appropriate filter and draw steps would be needed for the bundling algorithm selected. Figure 2 shows the relationship between our algorithmic choices at each step of the framework and the bundling quality metrics that we aim at optimizing.

`Filter`. The edges of E' for the skeleton subgraph are computed based on a weighting function. The edges that receive the highest weight will be included in the skeleton. The idea is to give a higher weight to those edges that are more "central" in the graph, based on the intuition that bundling against these edges optimizes the distortion.

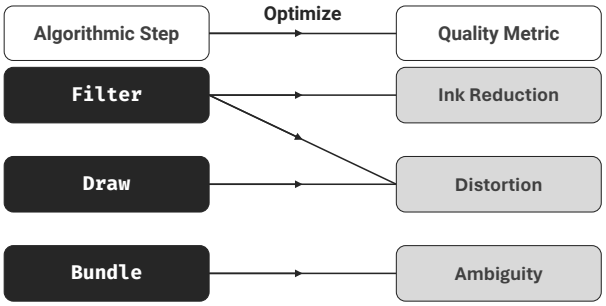

Figure 2 Relationships between our algorithmic choices at each step of the FDB framework and the bundling quality metrics that we aim at optimizing.

More formally, we weight the edges by two different methods. The first method uses the *edge betweenness* as a standard centrality measure for edges (see, e.g., [12]), defined as:

$$\text{EdgeBetweenness}(e) = \sum_{u,v \in V} \frac{\sigma(u, v \mid e)}{\sigma(u, v)}. \tag{4}$$

In the formula, $\sigma(u, v)$ denotes the number of shortest paths between any two vertices u and v of G, and $\sigma(u, v \mid e)$ is the number of such shortest paths passing through an edge e. The edge betweenness assigns a weight to each edge $e \in E$ based upon the computation of all-pairs shortest paths in G [3]. The weight of each edge is proportional to the number of shortest paths between all vertex pairs of G that pass through this particular edge.

A different approach, specifically tailored for the `Bundle` step, is to assign edge betweenness weights by considering the shortest paths only between those pairs of vertices that are actually adjacent in G, since these edges might later be bundled against their respective shortest path. We shall filter out those edges of G that appear in only a few such shortest paths. In the `Bundle` step, such edges will be reinserted one per time and possibly bundled against the edges of the skeleton. This leads to the following variant of the definition of edge betweenness, that we call *neighboring edge betweenness*:

$$\text{NeighboringEdgeBetweenness}(e) = \sum_{uv \in E} \frac{\sigma'(u, v \mid e)}{\sigma'(u, v)}, \tag{5}$$

where $\sigma'(u, v)$ is the number of shortest paths between u and v in $G_{uv} = (V, E \setminus \{uv\})$, and $\sigma'(u, v \mid e)$ is the number of these shortest paths passing through e. Algorithm 2 shows the pseudocode to compute the edge weights based on the neighboring edge betweenness. It is immediate to see that, since the edge betweenness of an unweighted graph can be computed in polynomial time (see, e.g., Brandes [3]), also the neighboring edge betweenness can be computed in polynomial time.

Once the edges are weighted, the skeleton G' is formed by filtering out the lighter edges from G. This can be done, for example, by computing a t-spanner [1] using the inverse weights. Recall that a t-spanner of a weighted graph G is a subgraph G' of G consisting of all the vertices of G and of a subset of its edges. The t-spanner property, for some $t > 1$, says that for each edge uv in G with weight $w(uv)$, there must be a (shortest) path π between u and v in G' whose weight $w(\pi) = \sum_{e \in \pi} w(e) \leq t \cdot w(uv)$. A typical approach to compute a t-spanner is to order the edges of the graph by increasing weight; starting from

Algorithm 2 Edge weights computation with neighboring edge betweenness.

Input: Graph $G = (V, E)$.
Output: Function $weights\colon E \to \mathbb{R}^+$.
for $uv \in E$ **do**
  $sp(uv) \leftarrow \text{shortestPaths}(u, v, G_{uv} = (V, E \setminus uv))$
for $e \in E$ **do**
  for $uv \in E$ **do**
    $sum \leftarrow 0$
    for $p \in sp(uv)$ **do**
      if $p.contains(e)$ **then**
        $sum \leftarrow sum + 1$
    $weights(e) \leftarrow weights(e) + \frac{sum}{|sp(uv)|}$
return $weights$

a graph $G' = (V, \emptyset)$, the edges are processed one by one in that order, and at each step it is decided whether the edge is inserted into G' or not; refer to Algorithm 3, where we use $w(e) = 1/weights(e)$.

Algorithm 3 Skeleton computation as a t-spanner [1, 38].

Input: Graph $G = (V, E)$, edge-weights $w\colon E \to \mathbb{R}^+$, value $t > 1$.
Output: t-spanner $G' = (V, E' \subseteq E)$.
$E' \leftarrow \emptyset$
$G' \leftarrow (V, E')$
$sortedEdges \leftarrow \text{sortAscending}(E, w)$
for $uv \in sortedEdges$ **do**
  $p \leftarrow \text{shortestPath}(G', u, v)$
  if p *is undefined* **then**
    $E' \leftarrow E' \cup \{uv\}$
  else
    $p.weight \leftarrow 0$
    for $e \in p$ **do**
      $p.weight \leftarrow p.weight + w(e)$
    if $p.weight > t * w(uv)$ **then**
      $E' \leftarrow E' \cup \{uv\}$
return $G' = (V, E')$

According to this algorithm, the lighter edges, i.e., those with the highest (neighboring) betweenness score, are more likely to be part of G' than the heavier edges, which have low betweenness. The higher we choose the value of t the sparser the skeleton G' will be, since it is less likely during the construction algorithm that there is no path of weight at most $t \cdot w(uv)$ in G'. On the one hand, this is expected to influence the ink ratio of the bundled layout. The intuition is that the sparser the skeleton, the lower the ink ratio. Namely, when the skeleton is sparser more edges of $E \setminus E'$ are bundled against the same portion of the skeleton. On the other hand, sparser skeletons and higher values of t in the spanner construction are expected to also lead to higher distortions in the final bundled drawings. Therefore, we expect that the choice of the value of t is very influential for the final bundling quality, and it should be balanced to achieve both good ink ratio and good distortion.

`Draw`. The goal of this step is to compute a drawing of the (unweighted) skeleton G' such that the graph-theoretic distance between its vertices is well reflected by their Euclidean distance in the layout. The intuition behind this choice is that such a layout would, in fact, give rise to bundled drawings with small distortion because it tends to straighten those paths against which edges are bundled. To this aim, we compute the output of the `Layout` step by means of a stress majorization approach with stochastic gradient descent (SGD) [11, 40].

`Bundle`. This step adds to the layout of the skeleton $G' = (V, E')$ those edges that are not part of it, i.e., edges in $E \setminus E'$. Let Γ be the layout of the skeleton computed in the previous step and let $uv \in E \setminus E'$ be a *non-skeletal edge*. We incrementally add each non-skeletal edge to Γ by adapting the Edge-Path bundling approach proposed in [37, 38], as described in Algorithm 4. More precisely, for each non-skeletal edge uv we consider the path in the skeleton between u and v that minimizes the sum of the Euclidean lengths of its edges. Let π be such a path: we bundle uv to π if the distortion is not above a predefined threshold $\delta > 1$; otherwise, uv is added to Γ as a straight-line unbundled segment. In our implementation we use the threshold $\delta = t$, for the same value of t applied in the `Filter` step. The rationale behind adapting the `EPB` approach to our scenario is that it avoids independent edge ambiguities by design. Indeed, an edge uv can only be bundled against a path of the skeleton connecting u to v. Moreover, we ensure that edges can only be bundled against a path if their distortion is not above the selected distortion threshold δ. Finally, bundling always has the goal of saving ink, thus improving on the ink ratio score.

Algorithm 4 Spanner-Edge-Path bundling algorithm [38].

Input: Graph $G = (V, E)$, skeleton $G' = (V, E')$, drawing Γ of G', maximum distortion $\delta > 1$.
Output: Control points for the edges in $E \setminus E'$.
for $uv \in E \setminus E'$ **do**
  $p \leftarrow \text{shortestPath}(G', \Gamma, u, v)$
  $p.length \leftarrow 0$
  for $xy \in p$ **do**
    /* $d(x, y)$ is the Euclidean distance between x and y in Γ */
    $p.length \leftarrow p.length + d(x, y)$
  if $p.length \leq \delta * d(u, v)$ **then**
    /* bundle edge uv against p */
    $controlPoints(uv) \leftarrow p.\text{getVertexCoordinates}()$
  else
    /* draw edge uv unbundled */
    $controlPoints(uv) \leftarrow (\Gamma(u), \Gamma(v))$
return $controlPoints$

4 Experimental Study

In this section, we present an experimental analysis that compares the bundling performances of four algorithms. We first recall these algorithms (Section 4.1); we then discuss our experimental hypotheses (Section 4.2); next we describe the data set and give implementation details (Section 4.3); finally, we present a statistical analysis of the results (Section 4.4).

4.1 Competing Algorithms

In the experiment, we consider the following four algorithms:

- `EB-FDB`: This algorithm is the first instantiation of the `FDB` framework described in Section 3.2.1, where the filtering step uses edge betweenness.
- `NEB-FDB`: This algorithm is the second instantiation of the `FDB` framework described in Section 3.2.1, where the filtering step uses neighboring edge betweenness.
- `ConfluentDrawing`: This algorithm creates a confluent-like drawing [2, 41] using the implementation of Zheng et al. [41].
- `PP-Bundling`: This algorithm computes a drawing of the input graph using standard stress majorization [11,40], and then it performs the Spanner Edge-Path bundling approach [38] (SEPB) on the computed drawing. It can be regarded as a representative of the common post-processing approach where the visual clutter is reduced by bundling edges on a given drawing. Hence, the graph spanner which is used for the bundling is computed on the Euclidean distances of a given drawing.

4.2 Experimental Hypotheses

We conceptually cluster our hypotheses into two groups. The first group is about the relationship between the `FDB` framework and the state-of-the-art bundling methods `ConfluentDrawing` and `PP-Bundling`, whereas the second group is about the direct comparison of the two instantiations `EB-FDB` and `NEB-FDB` of the framework.

The following hypotheses belong to the first group:

- **H1.1:** Algorithms `EB-FDB` and `NEB-FDB` perform better than the `PP-Bundling` algorithm in terms of ink ratio, distortion, and ambiguity.
- **H1.2:** Algorithms `EB-FDB` and `NEB-FDB` perform better than the `ConfluentDrawing` algorithm in terms of ink ratio, distortion, and ambiguity.

The following hypotheses belong to the second group:

- **H2.1:** Increasing the value of t in the t-spanner reduces the ink ratio of the bundled drawings computed with either the `EB-FDB` algorithm or the `NEB-FDB` algorithm.
- **H2.2:** For a given value of t, the `Filter` step of the `NEB-FDB` algorithm yields sparser t-spanners than those computed by the `Filter` step of the `EB-FDB` algorithm.
- **H2.3:** The `NEB-FDB` algorithm computes bundled drawings with higher distortion but lower ink ratio than those computed by the `EB-FDB` algorithm.

Rationale. Concerning **H1.1**, we recall that in the `PP-Bundling` algorithm the position of the nodes is computed by a standard stress majorization method, agnostic of the bundling step, which is in fact applied as a post-processing. On the other hand, the `Filter` step of the `EB-FDB` and the `NEB-FDB` algorithms is designed such that the edges of the path against which the bundling is executed are selected based on the graph topology and placed on a central position in the layout. This, we believe, positively impacts both ink ratio and distortion, because the long edges tend to be bundled against sets of edges that are central in the layout, avoiding long curves that pass through peripheral parts of the layout. This also potentially leads to layouts with less ambiguity compared to `PP-Bundling`: routing long edges towards the center likely reduces edge crossings and, consequently, it could reduce the number of sharp crossing angles.

Regarding **H1.2**, the `ConfluentDrawing` algorithm introduces dummy nodes and edges to the original graph. While these guides help construct a confluent drawing, we believe they might negatively impact distortion and ink ratio. Dummy elements are not part of the

actual data, and curves routed around them might become unnecessarily long compared to those computed without them. Also, by strategically placing heavier edges centrally, FDB reduces the chances of them intersecting with many other edges in peripheral areas and this could lead to less visual clutter and potentially an ambiguity reduction.

As for **H2.1**, higher values of t give rise to sparser skeletons. Therefore, by increasing t the number of edges against which the non-skeletal edges are bundled decreases, potentially leading to a smaller number of occupied pixels in the bundled layout.

Hypothesis **H2.2** originates from the observation that the Filter step of NEB-FDB weights the edges by only considering shortest paths between pairs of adjacent vertices, which could give rise to a smaller number of "heavily weighted" edges to be included in the skeleton than those computed by EB-FDB. We note, however, that while we expect this hypothesis to hold in typical real-world data sets, it is easy to find small counterexamples, where this is not the case. The same intuition is behind **H2.3**, as a skeleton with fewer edges gives rise to bundled drawings that use fewer pixels; however, this may have a negative impact on the distortion.

4.3 Dataset and Implementation Details

Our benchmark consists of graphs having a social network structure [21] and generated using the well-established Stochastic Block Model (SBM) [15]. The dataset is divided into four graph-size ranges: 20-50 nodes, 50-100 nodes, 100-150 nodes, and 150-200 nodes. Within each range, graphs are further categorized into five density classes based on the average number of edges per node (edge density). These classes range from sparse (2-4 edges per node) to dense (greater than 10 edges per node). We generated five graphs for each combination of graph-size range and density class, resulting in a total of 100 graphs.

The implementation of the Filter step of EB-FDB algorithm leverages the well-known NetworkX library [13] to compute betweenness centrality (Equation 4), which is based on the algorithm by Brandes [3]. The neighboring edge betweenness of Equation (5) is implemented in Python. The Draw step for both the EB-FDB algorithm and the NEB-FDB algorithm uses the Stochastic Gradient Descent method to optimize the stress of a straight line drawing of the skeleton [40]. Finally, the Bundle step uses the implementation of the Faster Edge-Path Bundling algorithm by Wallinger et al. [38]. The main code is in Python with integrated C++ methods. Specifically, the Stochastic Gradient Descent and the NetworkX library make use of more efficient C++ routines. Due to the different implementation languages we did not focus on performance comparisions. Still, we measured wallclock runtime and report it for EB-FDB (mean = 0.36s, max=3.30s) and NEB-FDB (mean = 23.04s, max=335.61s). We ran our experiments on a compute cluster with an AMD EPYC 7402, 2.80GHz 24-core CPU but restricted the computation to one core and 32GB of RAM, essentially simulating a powerful desktop PC.

The data set and the code used for the experimental analysis are available at the following OSF repository: https://osf.io/g4xqw/.

4.4 Analysis of the Results

We analyzed ink ratio, distortion and ambiguity separately. In general, a Shapiro-Wilk test revealed that no condition-metric pair was normally distributed. Hence, we applied the Wilcoxon signed-rank for comparing two conditions or the Friedmann test for three or more conditions to test for statistical significance ($\alpha = 0.01$). If statistical significance was found, we applied Bonferroni-corrected pairwise t-tests. We report effect size with the the Common Language Effect Size (CLES), i.e., the probability that the score of a random sample from one distribution is greater than the one of a random sample from some other distribution.

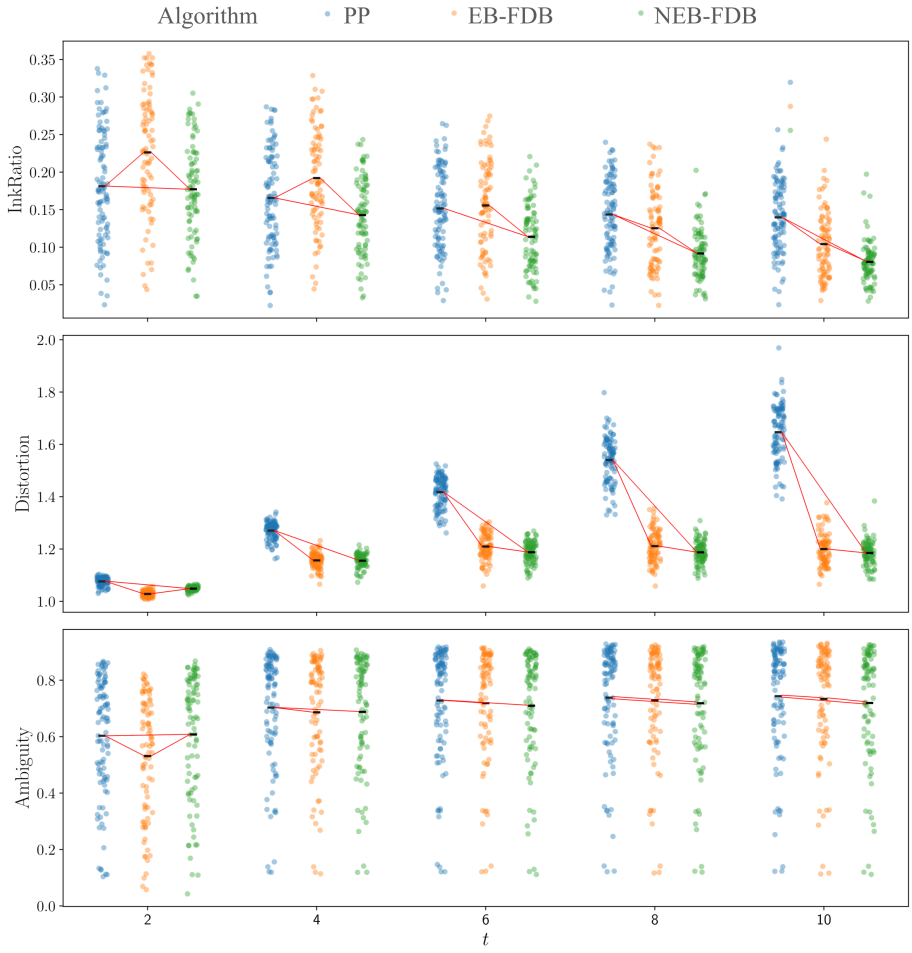

Figure 3 Strip plots of the experimental data about **H1.1**. The x-axis reports different t values. The y-axis reports: (top) ink ratio; (middle) distortion, and (bottom) ambiguity. Red segments connecting the mean values indicate statistically significant differences.

4.4.1 H1.1

Recall that the hypothesis is about `EB-FDB` and `NEB-FDB` outperforming `PP-Bundling` in ink ratio, distortion, and ambiguity. Our results are as follows (see also Figure 3). Note that the reported effect size is stated in terms of aggregated t, while individual levels can potentially have different effect sizes.

Ink Ratio. When aggregating all instances over t we partially confirm the hypothesis. `NEB-FDB` (mean value = 1.12) has lower ink ratio than `PP-Bundling` (mean value = 1.16) (CLES = 0.67, $p < 0.01$). However, there is no statistically significant difference between `EB-FDB` (mean value = 1.16) and `PP-Bundling`. If we compare different levels of t, then there is no significant differences for $t = 6$ between `EB-FDB` and `PP-Bundling`. For all other levels of t we have statistically significant results between the algorithms. Interestingly, with increasing t `EB-FDB` exhibits higher ink ratio reduction than `PP-Bundling`.

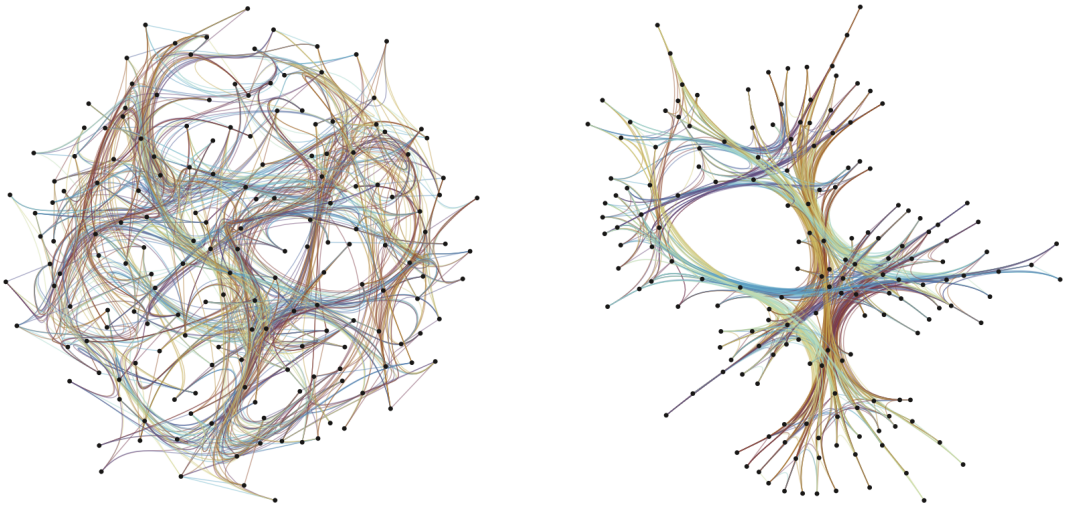

Figure 4 Instance with $|V| = 156$ and $|E| = 815$ from the benchmark dataset visualized with `PP-Bundling` (left) and `NEB-FDB` (right) with $t = 6$. The ink ratio is 0.21 for `PP-Bundling` and 0.13 for `NEB-FDB`; the mean distortion is 1.52 for `PP-Bundling` and 1.18 for `NEB-FDB`; the ambiguity is 0.91 for `PP-Bundling` and 0.90 for `NEB-FDB`.

Distortion. When aggregating t the results for distortion indicate that there is statistically significant difference between `PP-Bundling` (mean value = 1.39) compared to `EB-FDB` (mean value = 1.14) (CLES = 0.84, $p < 0.01$) and `NEB-FDB` (mean value = 1.13) (CLES = 0.85, $p < 0.01$). Furthermore, if we analyse individual values of t we find that there is positive correlation between t and distortion for `PP-Bundling`. In contrast, the mean distortion does not exhibit this behavior for `EB-FDB` or `NEB-FDB`.

Ambiguity. Again, if we aggregate instances over t the results indicate significant differences between `PP-Bundling` (mean value = 0.70) compared to `EB-FDB` (mean value = 0.63) (CLES = 0.58, $p < 0.01$) and `NEB-FDB` (mean value = 0.64) (CLES = 0.57, $p < 0.01$). Correlation analysis shows that for all three algorithms ambiguity correlates with increasing t.

Summary. For distortion and ambiguity our results support **H1.1**. However, for ink ratio we found support only for values of $t \neq 6$, i.e., for `EB-FDB` the ink ratio is higher for lower values of t but decreases faster with increasing t compared to the other algorithms. Also, with increasing t the distortion for `EB-FDB` and `NEB-FDB` remains somewhat constant while `PP-Bundling` increases.

Figure 4 shows two drawings of the same graph. The one to the left is computed by `PP-Bundling` while the one to the right is computed by `NEB-FDB`; in both cases $t = 6$.

4.4.2 H1.2

Recall that the hypothesis is about `EB-FDB` and `NEB-FDB` outperforming `ConfluentDrawing` in ink ratio, distortion, and ambiguity. Our results are as follows (see also Figure 5).

First, we decided on a value for t to have a fair comparison between `EB-FDB` and `NEB-FDB` against `ConfluentDrawing`. The choice for $t = 6$ was guided by the overall balanced performance regarding all three metrics.

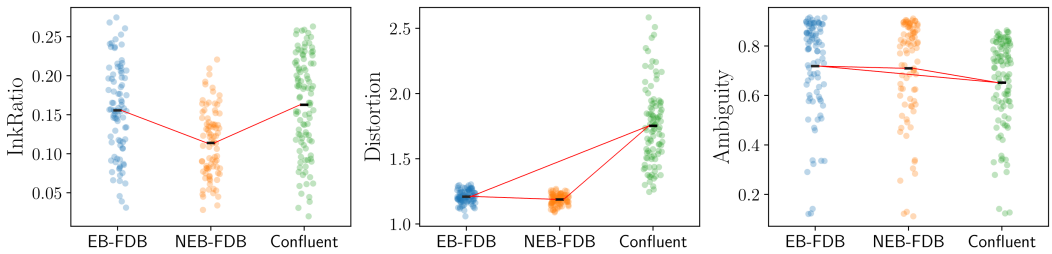

Figure 5 Strip plots of the experimental data about **H1.2**. The x-axis reports different algorithms.

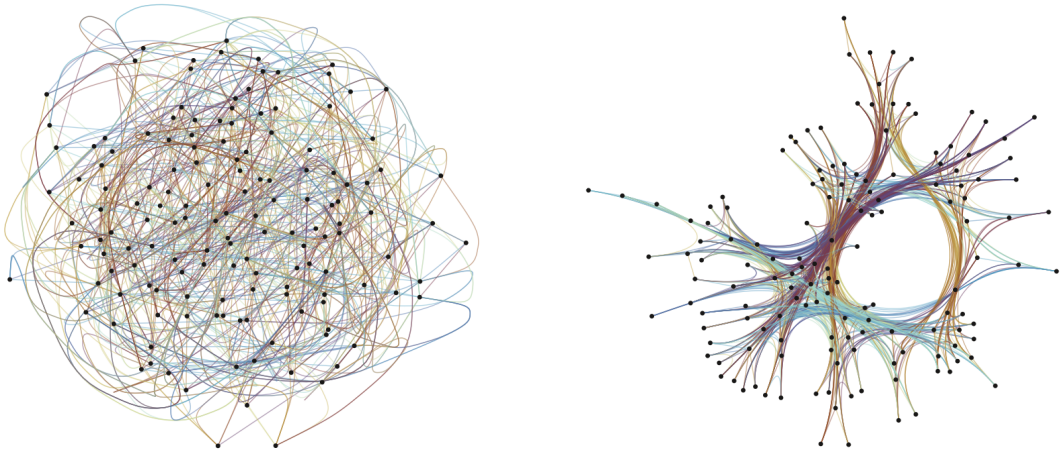

Figure 6 Instance with $|V| = 147$ and $|E| = 952$ from the benchmark dataset visualized with `ConfluentDrawing` (left) and `NEB-FDB` (right) with $t = 6$. The ink ratio is 0.23 for `ConfluentDrawing` and 0.09 for `NEB-FDB`; the mean distortion is 1.74 for `ConfluentDrawing` and 1.17 for `NEB-FDB`; the ambiguity is 0.82 for `ConfluentDrawing` and 0.89 for `NEB-FDB`.

Ink Ratio. Here, the ink ratio shows a significant difference for `ConfluentDrawing` (mean value = 0.16) when compared against `NEB-FDB` (mean value = 0.12) (CLES = 0.72, $p < 0.01$) but not for `EB-FDB` (mean value = 0.15).

Distortion. The comparison of the distortion values revealed that both `EB-FDB` (mean value = 1.17) (CLES > 0.99, $p < 0.01$) and `NEB-FDB` (mean value = 1.16) (CLES > 0.99, $p < 0.01$) have drastically less distortion than `ConfluentDrawing` (mean value = 1.75).

Ambiguity. Lastly, we found that `ConfluentDrawing` (mean value = 0.65) has lower ambiguity than `EB-FDB` (mean value = 0.67) (CLES = 0.58, $p < 0.01$) and `NEB-FDB` (mean value = 0.66) (CLES = 0.57, $p < 0.01$).

Summary. For ink ratio our results partially support **H1.2** and fully support it for distortion. **H1.2** is also supported for `NEB-FDB` with respect to mean distortion for all values of t that we considered. As for ambiguity, `ConfluentDrawing` outperforms both `NEB-FDB` and `EB-FDB`.

Figure 6 shows two drawings of the same graph. The one to the left is computed by `ConfluentDrawing` while the one to the right is computed by `NEB-FDB` with $t = 6$.

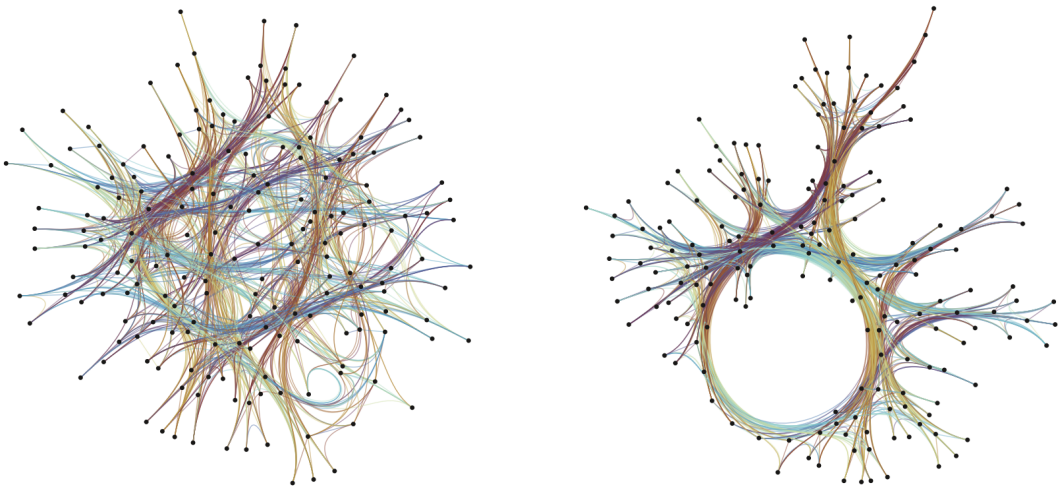

Figure 7 Instance with $|V| = 182$ and $|E| = 1268$ from the benchmark dataset visualized both with NEB-FDB, and with $t = 6$ (left) and $t = 10$ (right). The ink ratio is 0.16 in the drawing to the left and 0.09 in the drawing to the right.

4.4.3 H2.1

Recall that the hypothesis is that higher values of t yield lower ink ratio both with EB-FDB and with NEB-FDB.

Analyzing the ink ratio for different values of t individually for EB-FDB and NEB-FDB revealed a similar behavior. For both algorithms the ink ratio has no statistically significant differences between $t = 2$ and $t = 4$ and between $t = 8$ and $t = 10$. All other values had significant difference with effect sizes CLES < 0.4. This indicates that for $t \leq 4$ the spanner is dense and subsequently not many edges can be bundled. Similarly, for $t \geq 8$ the spanner is sparse and probably close to a spanning tree. Hence, the maximal amount of edges is bundled but there is no choice of path left when computing the bundling.

Summary. Our results support **H2.1** except when comparing $t = 2$ with $t = 4$ and when comparing $t = 8$ with $t = 10$. Figure 7 shows two drawings of the same graph computed by NEB-FDB for different t values.

4.4.4 H2.2

Recall that the hypothesis is about NEB-FDB using sparser t-spanners than EB-FDB. Refer to Figure 8 for our results.

Here we counted the edges in the spanner after computing it in the filter step of EB-FDB and NEB-FDB. The statistical analysis showed that instances computed with NEB-FDB have sparser spanners than EB-FDB (CLES $= 0.44$, $p < 0.01$). Furthermore, the effect size becomes stronger if we exclude higher values of t, i.e., all instances that are probably for both a spanning tree already.

Summary. Our results support **H2.2**.

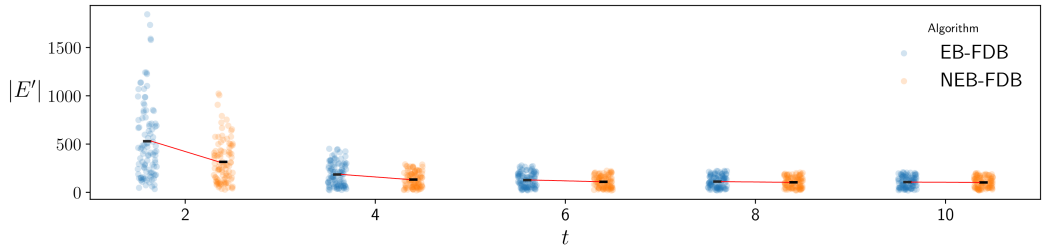

Figure 8 Strip plots presenting the impact of the parameter t on the number of edges in the spanners for EB-FDB and NEB-FDB.

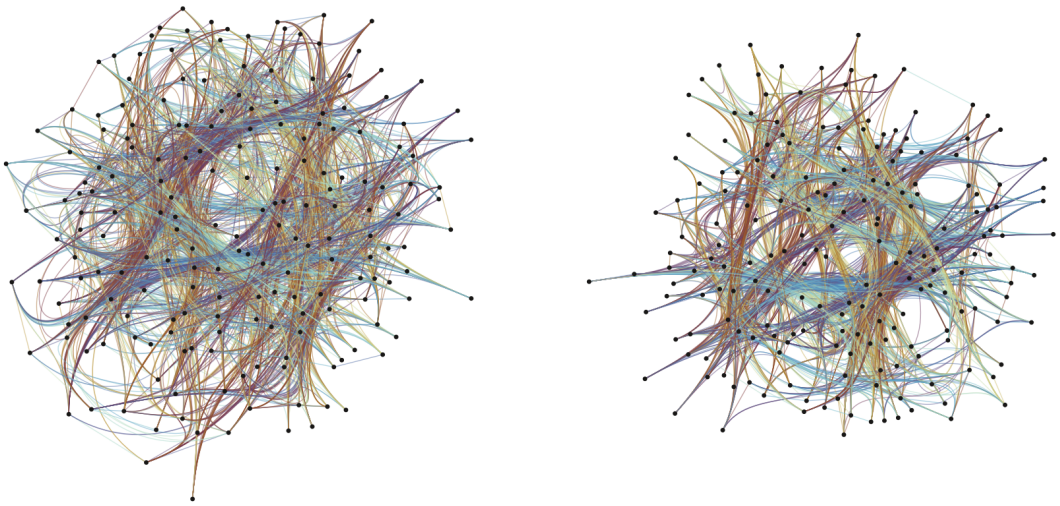

Figure 9 Instance with $|V| = 193$ and $|E| = 2614$ from the benchmark dataset visualized with EB-FDB (left) and NEB-FDB (right) with $t = 6$. The ink ratio is 0.24 in the drawing to the left and 0.21 in the drawing to the right; the mean distortion is 1.28 in the drawing to the left and 1.26 in the drawing to the right.

4.4.5 H2.3

Recall that the hypothesis is about NEB-FDB computing drawings with higher distortion but lower ink ratio than EB-FDB. Refer to Figure 3 for our results.

In direct comparison between EB-FDB and NEB-FDB we analyzed that both, ink ratio (CLES $= 0.35$, $p < 0.01$) and distortion (CLES $= 0.48$, $p < 0.01$) are higher for EB-FDB. Even though the result for distortion is statistically significant, the effect size only shows minor impact of the algorithm in the `Filter` step.

Summary. Our results partially support **H2.3**. Namely, NEB-FDB outperforms EB-FDB both in distortion and in ink ratio for the same values of t. Recall that, for a fixed value of t, the spanner computed for NEB-FDB is generally sparser than the spanner for EB-FDB. In contrast to our initial intuition, EB-FDB gives rise to higher distortion than NEB-FDB, since the curves representing the bundled edges are longer.

Figure 9 shows two drawings of the same graph. The one to the left is computed by EB-FDB while the one to the right is computed by NEB-FDB. In both cases with $t = 6$.

5 Final Remarks and Future Work

We conclude the paper addressing the lessons that we learned, pointing out to the limitations of our study, and discussing future research directions.

Lessons Learned. This study suggests that bundled drawing quality can be significantly enhanced by designing algorithms that directly optimize core quality metrics during the layout computation process, rather than relying solely on post-processing techniques. To this aim, we can use an algorithmic framework consisting of three main steps: `Filter`, `Draw`, and `Bundle`. The `Filter` step sparsifies the graph by identifying relevant edges, the `Draw` step computes a layout of the sparsified graph, and the `Bundle` step reinserts the missing edges into the layout. We showed that the computation of the sparsified graph can have significant impact on relevant bundling qualities. In particular, we weighted the edges of the graph and then applied a t-spanner. Our experiments have shown that both the edge weighting method and the choice of the parameter t in the t-spanner can influence the ink ratio, distortion, and ambiguity of the final drawing. More precisely, a value of t between 4 and 8, and in particular a value of $t = 6$, together with a weighting function based on neighboring edge betweenness, i.e., `NEB-FDB`, seems to be the most effective instantiation of the `FDB` framework in practice.

Limitations. We performed an experimental study that compares the `EB-FDB`, the `NEB-FDB`, the `PP-Algorithm`, and the `ConfluentDrawing` bundling algorithms on a data set consisting of 100 graphs having different numbers of nodes and densities. Although our results support our hypotheses in most cases, experiments involving additional datasets should be conducted. For example, one could consider a higher number of networks, having different sizes, and that are generated with algorithms different from SBM. Also, it would be interesting to evaluate our approach with real-world networks. At this point, the quality comparisons of the different bundled drawings are performed purely on the three quantitative measures of ink ratio, distortion, and ambiguity. Similarly, other drawing quality metrics should be tested to provide new perspectives on bundling performance.

Future Work. We studied two different algorithmic instances of the `FDB` framework, namely the `EB-FDB` algorithm and the `NEB-FDB` algorithm. We believe that the potential of the different steps of the `FDB` framework can be further investigated by considering different methods for each of the steps in the framework. It would be interesting to: (*i*) test different sparsification techniques such as, for example, the Simmelian backbone [28]; (*ii*) adopt different layout strategies for the drawing of the sparsified graph such as, for example, faithful force-directed methods [25]; (*iii*) test approaches that consider crossing angles in the `Bundle` step of which examples exist in the literature [5, 8]. Currently, we investigated the problem from a practical perspective, however, a complementary theoretical investigation into combining layout and bundling algorithms would be beneficial for further understanding. Also, it would be interesting to compare the drawings produced by `EB-FDB`, `NEB-FDB`, `PP-Algorithm`, and `ConfluentDrawing` in terms of aesthetic criteria such as, for example, angular resolution, crossing angle, and number of crossings. Finally, it is an open question to evaluate whether the optimized parameters are good proxies for graph readability.

References

1 Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discret. Comput. Geom.*, 9:81–100, 1993. doi:10.1007/BF02189308.

2 Benjamin Bach, Nathalie Henry Riche, Christophe Hurter, Kim Marriott, and Tim Dwyer. Towards unambiguous edge bundling: Investigating confluent drawings for network visualization. *IEEE Trans. Vis. Comput. Graph.*, 23(1):541–550, 2017. doi:10.1109/TVCG.2016.2598958.

3 Ulrik Brandes. A faster algorithm for betweenness centrality. *J. Math. Sociol.*, 25(2):163–177, 2001. doi:10.1080/0022250X.2001.9990249.

4 Weiwei Cui, Hong Zhou, Huamin Qu, Pak Chung Wong, and Xiaoming Li. Geometry-based edge clustering for graph visualization. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1277–1284, 2008. doi:10.1109/TVCG.2008.135.

5 Emilio Di Giacomo, Walter Didimo, Luca Grilli, Giuseppe Liotta, and Salvatore Agostino Romeo. Heuristics for the maximum 2-layer RAC subgraph problem. *Comput. J.*, 58(5):1085–1098, 2015. doi:10.1093/COMJNL/BXU017.

6 Matthew Dickerson, David Eppstein, Michael T. Goodrich, and Jeremy Y. Meng. Confluent drawings: Visualizing non-planar diagrams in a planar way. *J. Graph Algorithms Appl.*, 9(1):31–52, 2005. doi:10.7155/jgaa.00099.

7 Matthew Dickerson, David Eppstein, Michael T. Goodrich, and Jeremy Yu Meng. Confluent drawings: Visualizing non-planar diagrams in a planar way. In Giuseppe Liotta, editor, *Graph Drawing (GD'03)*, volume 2912 of *LNCS*, pages 1–12. Springer, 2003. doi:10.1007/978-3-540-24595-7_1.

8 Walter Didimo, Giuseppe Liotta, and Salvatore Agostino Romeo. A graph drawing application to web site traffic analysis. *J. Graph Algorithms Appl.*, 15(2):229–251, 2011. doi:10.7155/JGAA.00224.

9 Ozan Ersoy, Christophe Hurter, Fernando Vieira Paulovich, Gabriel Cantareiro, and Alexandru C. Telea. Skeleton-based edge bundling for graph visualization. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2364–2373, 2011. doi:10.1109/TVCG.2011.233.

10 Emden R. Gansner, Yifan Hu, Stephen C. North, and Carlos Eduardo Scheidegger. Multilevel agglomerative edge bundling for visualizing large graphs. In *Pacific Visualization Symposium (PacificVis'11)*, pages 187–194. IEEE, 2011. doi:10.1109/PACIFICVIS.2011.5742389.

11 Emden R. Gansner, Yehuda Koren, and Stephen C. North. Graph drawing by stress majorization. In János Pach, editor, *Graph Drawing (GD'04)*, volume 3383 of *LNCS*, pages 239–250. Springer, 2004. doi:10.1007/978-3-540-31843-9_25.

12 Michelle Girvan and Mark E. J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci.*, 99(12):7821–7826, 2002. doi:10.1073/pnas.122653799.

13 Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gäel Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Python in Science Conference (SciPy'08)*, pages 11–15, 2008.

14 Michael Hirsch, Henk Meijer, and David Rappaport. Biclique edge cover graphs and confluent drawings. In *Graph Drawing (GD'06)*, volume 4372 of *LNCS*, pages 405–416. Springer, 2007. doi:10.1007/978-3-540-70904-6_39.

15 Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Soc. Netw.*, 5(2):109–137, 1983.

16 Danny Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Trans. Vis. Comput. Graph.*, 12(5):741–748, 2006. doi:10.1109/TVCG.2006.147.

17 Danny Holten and Jarke J. van Wijk. Force-directed edge bundling for graph visualization. *Comput. Graph. Forum*, 28(3):983–990, 2009. doi:10.1111/j.1467-8659.2009.01450.x.

18 Weidong Huang, Peter Eades, and Seok-Hee Hong. A graph reading behavior: Geodesic-path tendency. In *Pacific Visualization Symposium (PacificVis'09)*, pages 137–144. IEEE, 2009. doi:10.1109/PACIFICVIS.2009.4906848.

19 Christophe Hurter, Ozan Ersoy, and Alexandru Telea. Graph bundling by kernel density estimation. *Comput. Graph. Forum*, 31(3):865–874, 2012. doi:10.1111/j.1467-8659.2012.03079.x.

20 Antoine Lambert, Romain Bourqui, and David Auber. Winding Roads: Routing edges into bundles. *Comput. Graph. Forum*, 29(3):853–862, 2010. doi:10.1111/j.1467-8659.2009.01700.x.

21 Clement Lee and Darren J. Wilkinson. A review of stochastic block models and extensions for graph clustering. *Appl. Netw. Sci.*, 4(1):122, 2019. doi:10.1007/S41109-019-0232-2.

22 Antoine Lhuillier, Christophe Hurter, and Alexandru C. Telea. FFTEB: edge bundling of huge graphs by the fast fourier transform. In *Pacific Visualization Symposium (PacificVis'17)*, pages 190–199. IEEE, 2017. doi:10.1109/PACIFICVIS.2017.8031594.

23 Antoine Lhuillier, Christophe Hurter, and Alexandru C. Telea. State of the art in edge and trail bundling techniques. *Comput. Graph. Forum*, 36(3):619–645, 2017. doi:10.1111/cgf.13213.

24 Sheng-Jie Luo, Chun-Liang Liu, Bing-Yu Chen, and Kwan-Liu Ma. Ambiguity-free edge-bundling for interactive graph visualization. *IEEE Trans. Vis. Comput. Graph.*, 18(5):810–821, 2012. doi:10.1109/TVCG.2011.104.

25 Amyra Meidiana, Seok-Hee Hong, and Peter Eades. Shape-faithful graph drawings. In Patrizio Angelini and Reinhard von Hanxleden, editors, *Graph Drawing and Network Visualization (GD'22)*, volume 13764 of *LNCS*, pages 93–108. Springer, 2022. doi:10.1007/978-3-031-22203-0_8.

26 Quan Hoang Nguyen, Peter Eades, and Seok-Hee Hong. On the faithfulness of graph visualizations. In *Pacific Visualization Symposium (PacificVis'13)*, pages 209–216. IEEE, 2013. doi:10.1109/PacificVis.2013.6596147.

27 Quan Hoang Nguyen, Seok-Hee Hong, and Peter Eades. TGI-EB: A new framework for edge bundling integrating topology, geometry and importance. In *Graph Drawing (GD'11)*, volume 7034 of *LNCS*, pages 123–135. Springer, 2011. doi:10.1007/978-3-642-25878-7_13.

28 Bobo Nick, Conrad Lee, Pádraig Cunningham, and Ulrik Brandes. Simmelian backbones: amplifying hidden homophily in facebook networks. In Jon G. Rokne and Christos Faloutsos, editors, *Advances in Social Networks Analysis and Mining (ASONAM'13)*, pages 525–532. ACM, 2013. doi:10.1145/2492517.2492569.

29 Sergey Pupyrev, Lev Nachmanson, Sergey Bereg, and Alexander E. Holroyd. Edge routing with ordered bundles. In Marc J. van Kreveld and Bettina Speckmann, editors, *Graph Drawing (GD'11)*, volume 7034 of *LNCS*, pages 136–147. Springer, 2011. doi:10.1007/978-3-642-25878-7_14.

30 Sergey Pupyrev, Lev Nachmanson, Sergey Bereg, and Alexander E. Holroyd. Edge routing with ordered bundles. *Comput. Geom.*, 52:18–33, 2016. doi:10.1016/j.comgeo.2015.10.005.

31 David Selassie, Brandon Heller, and Jeffrey Heer. Divided edge bundling for directional network data. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2354–2363, 2011. doi:10.1109/TVCG.2011.190.

32 Alexandru Telea and Ozan Ersoy. Image-based edge bundles: Simplified visualization of large graphs. *Comput. Graph. Forum*, 29(3):843–852, 2010. doi:10.1111/j.1467-8659.2009.01680.x.

33 Naoko Toeda, Rina Nakazawa, Takayuki Itoh, Takafumi Saito, and Daniel Archambault. Convergent drawing for mutually connected directed graphs. *J. Vis. Lang. Comput.*, 43:83–90, 2017. doi:10.1016/j.jvlc.2017.09.004.

34 Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

35 Matthew van der Zwan, Valeriu Codreanu, and Alexandru C. Telea. CUBu: universal real-time bundling for large graphs. *IEEE Trans. Vis. Comput. Graph.*, 22(12):2550–2563, 2016. doi:10.1109/TVCG.2016.2515611.

36 Markus Wallinger. FDB Framework. Software (visited on 2024-09-13). doi:10.17605/OSF.IO/G4XQW.

37 Markus Wallinger, Daniel Archambault, David Auber, Martin Nöllenburg, and Jaakko Peltonen. Edge-path bundling: A less ambiguous edge bundling approach. *IEEE Trans. Vis. Comput. Graph.*, 28(1):313–323, 2022. doi:10.1109/TVCG.2021.3114795.

38 Markus Wallinger, Daniel Archambault, David Auber, Martin Nöllenburg, and Jaakko Peltonen. Faster edge-path bundling through graph spanners. *Comput. Graph. Forum*, 42(6), 2023. doi:10.1111/CGF.14789.

39 Jieting Wu, Lina Yu, and Hongfeng Yu. Texture-based edge bundling: A web-based approach for interactively visualizing large graphs. In *Big Data (BigData'15)*, pages 2501–2508. IEEE, 2015. doi:10.1109/BigData.2015.7364046.

40 Jonathan X. Zheng, Samraat Pawar, and Dan F. M. Goodman. Graph drawing by stochastic gradient descent. *IEEE Trans. Vis. Comput. Graph.*, 25(9):2738–2748, 2019. doi:10.1109/TVCG.2018.2859997.

41 Jonathan X. Zheng, Samraat Pawar, and Dan M. Goodman. Further towards unambiguous edge bundling: Investigating power-confluent drawings for network visualization. *IEEE Trans. Vis. Comput. Graph.*, 27(03):2244–2249, 2021. doi:10.1109/TVCG.2019.2944619.