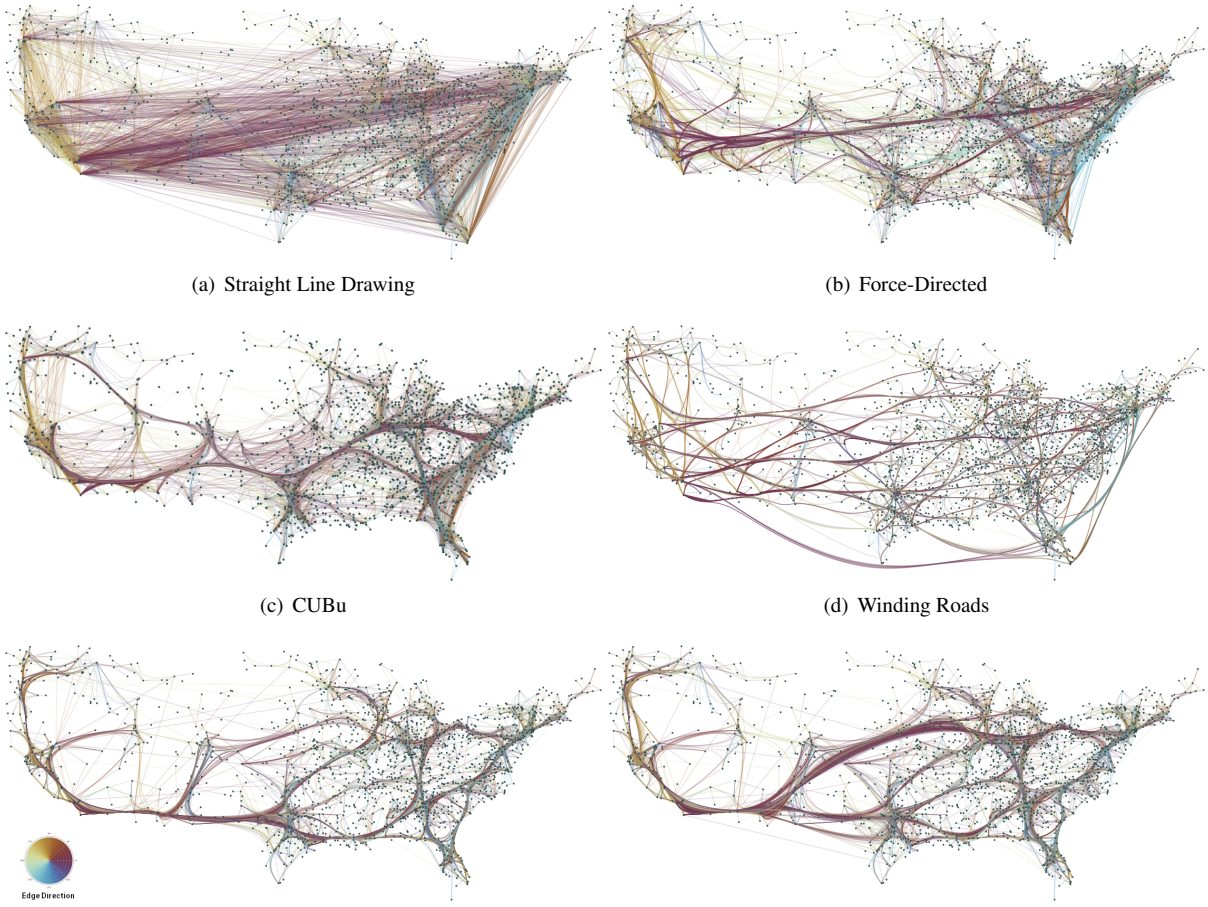
Edge-Path Bundling: A Less Ambiguous Edge Bundling Approach

Markus Wallinger* TU Wien
Daniel Archambault† Swansea University
David Auber‡ University of Bordeaux
Martin Nöllenburg* TU Wien
Jaakko Peltonen§ Tampere University



(a) Straight Line Drawing (b) Force-Directed

(c) CUBu (d) Winding Roads

(e) Edge-Path Bundling (f) Edge-Path Bundling Directed

Fig. 1. Edge bundling of the Migrations dataset. **(a)** Straight line drawing. **(b)** Force-Directed edge bundling aggregates edges well, but overaggregates at the centre of the drawing making it difficult to see patterns in the east-west flow (red bundle). **(c)** CUBu has a similar drawback at the centre of the map to a lesser degree. **(d)** Winding Roads divides this structure into several smaller flows, but they may not be necessarily related to graph structure. **(e)** Edge-Path bundling is able to distinguish between several flows that reflect paths in the underlying graph. **(f)** When edge direction is considered, the algorithm is able to further subdivide these flows based on direction.

Abstract—Edge bundling techniques cluster edges with similar attributes (i.e. similarity in direction and proximity) together to reduce the visual clutter. All edge bundling techniques to date implicitly or explicitly cluster groups of individual edges, or parts of them, together based on these attributes. These clusters can result in ambiguous connections that do not exist in the data. Confluent drawings of networks do not have these ambiguities, but require the layout to be computed as part of the bundling process. We devise a new bundling method, Edge-Path bundling, to simplify edge clutter while greatly reducing ambiguities compared to previous bundling techniques. Edge-Path bundling takes a layout as input and clusters each edge along a weighted, shortest path to limit its deviation from a straight line. Edge-Path bundling does not incur independent edge ambiguities typically seen in all edge bundling methods, and the level of bundling can be tuned through shortest path distances, Euclidean distances, and combinations of the two. Also, directed edge bundling naturally emerges from the model. Through metric evaluations, we demonstrate the advantages of Edge-Path bundling over other techniques.

*e-mail: {mwallinger, noellenburg}@ac.tuwien.ac.at
†e-mail: d.w.archambault@swansea.ac.uk
‡e-mail: auber@labri.fr
§e-mail: jaakko.peltonen@tuni.fi

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

1 Introduction

Since its introduction [24], edge bundling approaches cluster groups of individual edges together based on similar properties. The original paper required a hierarchy imposed on top of the network, but quickly techniques were developed to bundle edges with shared attributes [36] (e.g. proximity and movement in the same direction [15,25,31,46]) (see Fig. 1). However, all these approaches implicitly or explicitly cluster individual edges, or parts of them (such as their pixels), independent of graph structure. This causes a type of ambiguity shown in Fig. 2(a) called the *independent edge ambiguity*: two independent edges can be clustered together leading to the perception of false adjacencies that do not exist in the underlying graph.

Confluent drawings [16] considered a very similar problem, and cluster edges based on their participation in bicliques. Therefore, confluent drawings do not suffer from independent edge ambiguities. However, confluent drawings bundle graphs significantly less, leaving much edge clutter on-screen. Also, these approaches compute both the layout of the network and the bundling simultaneously. There have been a number of attempts to relax the strict constraints of confluent drawings to move towards unambiguous bundling [10, 57], but all approaches still require the layout to be computed alongside the bundling.

In all such approaches, groups of edges with similar attributes (typically, edge slope and proximity) or parts of them (i.e. their pixels) are bundled together if they are aligned well. However, when clustering these edges, the underlying graph structure is not fully considered. In Fig. 1, a straight line drawing and three popular bundling approaches are shown. Force-Directed edge bundling [25] and CUBu [50] greatly simplify the edge clutter clearly revealing the direction of flows. This is particularly visible in the east-west flows of the network (the red edges) at the centre of the drawing. However, unrelated edges can be pulled together into a bundle causing ambiguity in the patterns of connections. Grid-based techniques, such as Winding Roads [31], divide the edges into much smaller bundles, but these bundles can contain unrelated edges. The approach presented in this paper, Edge-Path bundling, bundles edges with shortest paths between their endpoints. Therefore, unrelated edges will not be bundled and all bundles reflect an underlying path in the graph. In the undirected version, it is now clear that the central flow divides into two: one that heads towards the great lakes region and another towards Texas. Further detail is also visible on the east coast and the great lakes region not visible in the other diagrams. Directed Edge-Path bundling reveals that these bundles actually consist of three main streams indicated in red. In the straight line drawing, these flows are somewhat visible, but the pattern is not revealed by any other technique except directed Edge-Path bundling; in particular, the flow from California to Texas divides into three.

This paper introduces a new approach to edge bundling that does not consider groups of individual edges, their pixels, or bicliques as the primitive for bundling. Instead, given an input layout (i.e. trail-sets [36]), it considers clustering edges with a weighted path as the primitive for bundling. Each long edge in the graph is bundled to a shortest path that exists between the endpoints of the edge (Fig. 2(b) and 2(c)). By definition, Edge-Path bundling does not suffer from independent edge ambiguities as a path must exist in the graph for the edge to be bundled, but it is far less restrictive than the rules imposed by confluent drawings. Also, the approach naturally expresses both undirected and directed edge bundling without modification to the algorithm. We demonstrate that the approach has more significant bundling when compared to confluent drawings while simultaneously eliminating independent edge ambiguities.

2 Related Work

Since the introduction of edge bundling by Holten [24] and confluent drawing by Dickerson *et al.* [16] for reducing edge clutter in graph drawings, the research area has been very active and many approaches have been devised by clustering edges, or their pixels, with similar properties together. The current state-of-the-art in edge bundling can incur independent edge ambiguities (Fig. 2(a)) when aggregating edges into clusters. Confluent drawings restrict the bundling to perfect bicliques to avoid this case, but often the imposed constraints are too strict for significant bundling to occur.

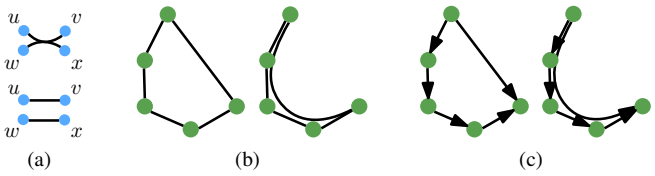


Fig. 2. Independent edge ambiguity and Edge-Path bundling. **(a)** Two disconnected edges can be bundled together in edge bundling approaches resulting in false connections between (u,x) and (v,w). **(b)** Edge-Path bundling avoids this issue by bundling long edges with weighted shortest paths. **(c)** For directed graphs, directed paths are used.

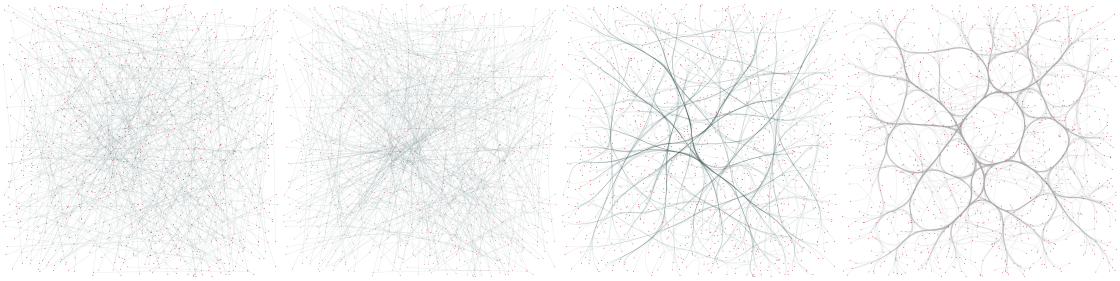
Edge-Path bundling devises a necessary rule to have efficient bundling while completely eliminating independent edge ambiguities. Intuitively, when a pair of disconnected edges (u,v) and (w,x) are bundled, a path can appear to exist between (u,x) and (v,w) where a connection does not exist at all. Also, Edge-Path bundling can avoid bundling patterns where there are none. In Fig. 3, independent edges are placed randomly in the cube and all bundling techniques find a pattern in this graph whereas Edge-Path bundling and confluent drawings do not. Edge-Path bundling does not have this issue as it only bundles edges with weighted paths in a particular layout. Although related to edge bundling, it is a fundamentally different approach that does not neatly fit into any of these categories.

Edge Bundling. Since the first techniques were described, many bundling techniques have been explored [36] often inspired by the work on flow maps [42]. A number of techniques have been proposed, but all techniques have one common primitive: clustering groups of edges together that share similar attributes.

Edge bundling was introduced to the visualisation community by Holten [24]. In his seminal work, a hierarchy was superimposed on top of the network, usually via a treemap variant [13, 30], and the centroids of the cluster hierarchy are used as control points to cluster edges into merging and splitting streams. Soon after, the requirement for a hierarchy was eliminated and replaced by a desired property whereby nearby edges headed in a similar direction with similar length are bundled together. A number of approaches were created with this idea including using a triangular mesh [15], grids and quad-trees [31, 37], force-directed algorithms [25, 40], force-directed algorithms with edge direction encoded in the force system with compatibility measures (such as connection distance) [45], sparse visibility spanners [43], and multilevel clustering [21]. Domain-specific clusters and layouts have been used to help with the bundling process [32, 44] as well as more general clusters [47]. Image-based [19, 29, 35, 46, 50, 54] techniques operate on the pixels of individual edges. These approaches create a density or similarity map computed at pixel level by summing up the contributions of all edges, after which all edges are independently advected upstream along gradients of the map. Bundling has also been used to simplify clusters in parallel coordinate plots [41].

Edge bundling techniques greatly reduce edge clutter by implicitly or explicitly clustering groups of edges or parts of them (i.e. their pixels) and allow visualisation of higher-level flow patterns in the network that otherwise would not be visible. However, all these approaches will suffer from the independent edge ambiguities to a degree. Also, as stated in the survey [36], random patterns can be created in data where there are no patterns. In this paper, we seek a compromise: efficient bundling that greatly simplifies the network while completely eliminating independent edge ambiguities. We accomplish this by bundling long edges with a weighted shortest path between their endpoints instead of clustering groups of edges together.

Confluent Drawings. Confluent drawings are visually similar to edge bundling but, by design, they do not suffer from independent edge ambiguities. The key idea of a confluent bundle is that only bicliques can be bundled, i.e., $K_{n,m}$ subgraphs. This guarantees that all connections implied by a bundle are actually present in the graph. In their orig-



(a) Edge-Path, Confluent (b) Force-directed (c) Winding Roads (d) CUBu

Fig. 3. Bundling of a noise graph. Randomly placed vertices ($|V| = 1000$) are connected by a perfect matching, resulting in $|E| = |V|/2$ disconnected edges. No structure is present in this graph and there should be no bundles (See [36] Figure 19). Force-directed, Winding Roads and CUBu give the impression of structure in the underlying data when there is none. Edge-Path bundling and confluent drawings do not find bundles.

inal and theoretically motivated definition, confluent drawings do not permit any edge crossings [16, 28] and as a consequence some graphs do not have a confluent drawing at all. Variations of confluent drawings, such as Δ-confluent [17] and strict confluent drawings [18, 20], have been studied from a theoretical point of view, showing mathematical characterizations and the NP-completeness of recognizing graphs that admit certain types of confluent drawings. While most results on confluent drawings do not provide algorithms or implementations, Dickerson *et al.* [16] and Hirsch *et al.* [23] present and implement some heuristics based on detecting cliques and bicliques in order to introduce confluent bundles. Confluent drawings are based on an abstract graph input and typically solve both the graph layout and the edge bundling in a combined way. Therefore, existing confluent drawing techniques do not readily apply to edge bundling of pre-embedded networks. In this paper, we devise a compromise that is not as strict as confluent layout, but more effectively bundles the network so that higher-level features become visible while reducing topological ambiguities.

Hybrid Approaches. A number of papers have used confluent drawing approaches as a basis and have relaxed some of the more restrictive constraints of the approach to achieve greater simplification. Bach *et al.* [10] explore ways to produce less ambiguous bundling by relaxing the strict planarity constraint of confluent drawings. Their approach computes a power graph decomposition and uses the resulting hierarchy to both draw and bundle the network. Zheng *et al.* [57] extend Bach *et al.* [10] to create strict, power-confluent drawings, based on additional constraints that further reduce crossings and ambiguities.

The two approaches described above were the first attempts to relax some of the constraints of confluent drawings to produce less ambiguous edge bundlings. In this paper, we introduce a new approach with a different primitive, but that finds a similar compromise: greater bundling while completely avoiding independent edge ambiguities.

Visualisations and Measures of Bundling Quality. Alongside bundling algorithms, significant work has been undertaken to devise metrics to evaluate how well bundling algorithms perform with respect to each other and in general. Metrics have been devised to measure faithfulness [38, 39], entropy [55], geodesic path tendency or distortion from the straight line distance [26], and data-ink ratio to quantify simplification [48]. We use and adapt these metrics in our evaluation.

Wang *et al.* [52] give methods to visualise ambiguities in graph layouts, including bundling. In this approach, alignment in edge direction and proximity, a classic definition for bundling suitability, are used to highlight ambiguities in the graph. Nguyen *et al.* define the notion of faithfulness in graph visualisation: *"the underlying network data and the visual representation are logically consistent"* [38]. Edge bundling ambiguities is used to illustrate faithfulness or lack of it in a representation. Our work, by definition, increases the faithfulness of bundling representation by avoiding certain types of ambiguity.

Summary. Edge bundling techniques improve graph readability, but suffer from independent edge ambiguities and can introduce patterns where none exist in the underlying data. Confluent drawings and hybrid approaches do not suffer from independent edge ambiguities, but have a lower degree of bundling and compute the drawing of the network as part of the bundling process. Thus, this paper introduces a new bundling primitive (edge to path) and a new bundling algorithm based on this primitive, that produces a more faithful bundling [38] of a layout.

3 Edge-Path Bundling Algorithm

Algorithm 1 presents the pseudocode for our approach. The method is surprisingly simple and requires only four parameters: the network $G = (V,E)$, a drawing D_G of G, a maximum distortion threshold k, and an edge weight factor d. The algorithm creates a local hashset, *lock*, that indicates when edges will be excluded from bundling by the algorithm. It also uses a second hashset *skip* which includes edges that will be skipped from shortest path calculations. Initially, all edges in both hashsets are false. The algorithm takes into account the length of an edge in D_G to determine its suitability for inclusion in a shortest path: shortest paths that result in shorter detours in Euclidean space are preferred. Therefore, a third hashset stores the weight of each edge which is the Euclidean edge length raised to a power d. The exponent d can be used to tune exclusion of short edges from the bundling.

As a first step, the edges are sorted in decreasing weight order and then processed in that order. Thus, our algorithm prefers to bundle long edges over short ones. If an edge is locked, the algorithm does not process it. Otherwise, it is excluded from shortest path computations and processed. The first stage of bundling is determining the shortest, weighted path between the endpoints of the edge, excluding the edge itself and previously bundled edges from this calculation. In order to compute this path, we use Dijkstra's algorithm to compute the shortest path that takes the minimum detour from straight line distance. If such a path does not exist or the detour is greater than a distortion threshold of k times the straight line distance, the edge is not bundled and it is reintroduced into shortest path calculations. Otherwise, the edge is bundled and the algorithm uses the vertices along the path in the drawing as control points for the edge. The bundled edge is excluded from all future shortest path calculations as it will be rendered using a curve. As all the edges along the path are now included in an Edge-Path bundle, these edges are locked. Edges along the path should not be bundled as they serve as the control points for one or more bundled edges in the graph, but they still can participate in shortest path calculations to bundle other edges. The final drawing of the network uses these control points to render each edge using a smooth Bézier curve. A smoothing parameter allows the algorithm to control the bundling strength in the final drawing by inserting additional control points along the path. A factor of one places control points on the vertices. Increasing this factor to two places additional control points on the centre of the straight line between two consecutive control points. Higher factors apply this rule recursively.

Overall, Edge-Path bundling has a worst case time complexity of $O(|E|^2 \log |V|)$ as the Dijkstra algorithm (priority queue heap implementation) runs $O(|E|)$ times. However, the distortion threshold k can be used to stop Dijkstra's algorithm once this threshold is exceeded, reducing the chance that this worst case complexity is observed.

Algorithm 1: Edge-Path Bundling Algorithm

Input : Graph $G = (V, E)$, input drawing D_G, maximum distortion k, edge weight factor d.
Output : Control points for an Edge-Path bundled drawing Γ.

```
for e ∈ E do
    lock(e) ← False
    skip(e) ← False
    weight(e) ← (D_G.edgeLength(e))^d
sortedEdges ← sortDescending (E, weight)
for e ∈ sortedEdges do
    if lock(e) then
        continue
    skip(e) ← True
    s ← source(e)
    t ← target(e)
    //Dijkstra excluding edges in skip from G
    p ← dijkstraAlgorithm (G, s, t, weight, skip)
    if p == null then
        skip(e) ← False
        continue
    if p.length() > k * D_G.edgeLength(e) then
        skip(e) ← False
        continue
    for m ∈ p do
        lock(m) ← True
    controlPoints(e) ← p.getVertexCoords()
return controlPoints
```

Fig. 4. Path endpoint ambiguity. A connection exists between (u,w) and (v,x) and there is a path between u and x. When edges are bundled along paths, the viewer could perceive a direct edge between (u,x) if the bundling is strong.

3.1 The Lesser Ambiguities of Edge-Path Bundling

Although our approach is free of independent edge ambiguities by definition, it is not completely free of ambiguities. We now describe the ambiguities incurred by Edge-Path bundling. It is important to note that the two ambiguities we define here can occur in all other edge bundling methods, but have not been concretely defined previously.

Path Endpoint Ambiguity. Edge-Path bundling does not create connections between vertices that do not exist in the underlying graph. However, it can produce ambiguities in edge endpoints along a path. Fig. 4 shows this ambiguity. A path exists, by definition between u and x, but it can be difficult to tell if there is a direct connection between u and x in the Edge-Path bundle, depending on bundling strength.

Edge Crossing Ambiguity. When two edges or bundles cross, there could be an ambiguity if the crossing angle is shallow [27]. This is a fundamental ambiguity of graph drawing in general, which also affects the unbundled straight-line input drawing D_G.

4 Quality Metrics

In order to evaluate the edge bundling created by different bundling algorithms, we make use of three quantitative metrics. They measure the amount of clutter reduction using the *ink reduction*, the *distortion* of edge lengths in the graph, and the amount of adjacency *ambiguity* in the bundled layout. This section defines these quality metrics. We remark that no single quality metric can fully judge a bundled layout, but we think that a good bundled layout, which reduces visual clutter but at the same time aims to be faithful to distances and graph topology, will do fairly well on all three metrics.

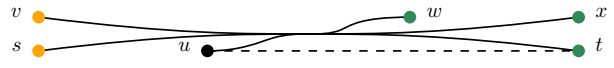


Fig. 5. Consider an edge e with endpoints (s,t). The bundling implies $N_\Gamma(s,e) = \{t,w,x\}$ and $N_\Gamma(t,e) = \{s,u,v\}$. Additionally, there exists an edge with endpoints (u,t) and therefore we would consider u a true neighbour of t resulting in $N^f_\Gamma(s,e) = \{w,x\}$ and $N^f_\Gamma(t,e) = \{v\}$

4.1 Ink Reduction

Let I be the grayscale bitmap image $I \in \{0,\dots,255\}^{m\times n}$ of a bundled graph layout Γ and let $I^B \in \{0,1\}^{m\times n}$ be the binarization of I such that

$$I^B(i,j) = \begin{cases} 1 & I(i,j) \geq \delta \\ 0 & I(i,j) < \delta \end{cases}, \tag{1}$$

where $0 \leq \delta \leq 255$ is a global gray value threshold to consider a pixel occupied. Similarly we define J as the unbundled input grayscale image. The ink-reduction of I with respect to J is defined as:

$$\text{ink}_J(I) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} I^B(i,j)}{\sum_{i=1}^{m}\sum_{j=1}^{n} J^B(i,j)}. \tag{2}$$

Under the assumption that the bundling reduces the number of pixels occupied by ink, the ink reduction takes a value between 0 and 1 and measures the factor by which the number of pixels occupied in the graph layout is reduced. An ink reduction close to 1 indicates a low degree of bundling, whereas smaller values indicate a higher degree of bundling. It is possible to obtain values larger than 1 if the bundling increases the number of pixels occupied.

4.2 Distortion

Let $e = (u,v) \in E$ be an edge in G with Euclidean length of $||u-v||$ and let $d_\Gamma(u,v)$ be the length of the curve connecting u and v in the bundled layout Γ. We define the distortion of layout Γ as the average distortion of its edges:

$$\text{dist}(\Gamma) = \frac{1}{|E|}\sum_{(u,v)\in E}\frac{d_\Gamma(u,v)}{||u-v||} \tag{3}$$

The distortion measures the factor by which length of an edge increases in the bundled layout on average. Distortion values near 1 mean that lengths of bundled edges remain close to the Euclidean distance of their endpoints; larger values mean edge bundles have longer detours from straight line distance, making adjacencies harder to read [26].

4.3 Ambiguity

We define an ambiguity metric, inspired by faithfulness [38], to measure the number and severity of perceivable false connections in a bundled layout (Fig. 5). Let $e = (s,t) \in E$ in a bundled layout Γ. We define the set of reachable neighbors of the endpoint s along e as $N_\Gamma(s,e) = \{v \in V \mid \exists$ ambiguous connection from s to v in $\Gamma\}$; analogously we define $N_\Gamma(t,e)$ for the endpoint t. An ambiguous connection occurs if another edge $e' = (u,v)$ intersects or is closer than a distance threshold ε at a point p on e in Γ and the angle at p between e and e' is smaller than a threshold θ. Intuitively, edges e and e', locally in p, are difficult to distinguish and thus the endpoint of e', say v, forming the small angle with t is a reachable neighbor in $N_\Gamma(s,e)$, whereas the other endpoint u belongs to $N_\Gamma(t,e)$, independently of (s,v) or $(u,t) \in E$ or not.

The reachable neighbor sets contain some true and false neighbors and we may take this classification based on a graph distance threshold $\delta \geq 1$: the set of true neighbors is defined as $N^t_\Gamma(s,e) = \{v \in N_\Gamma(s,e) \mid d_G(s,v) \leq \delta\}$ and the set of false neighbors as $N^f_\Gamma(s,e) = N_\Gamma(s,e) \setminus N^t_\Gamma(s,e)$. Here let $d_G(s,v)$ denote the hop distance between s and v in G, i.e., the length of the shortest unweighted path between s and v in G. We can now define the ambiguity of Γ:

$$\text{amb}(\Gamma) = \frac{\sum_{v\in V}\sum_{e=(v,w)\in E}|N^f_\Gamma(v,e)|}{\sum_{v\in V}\sum_{e=(v,w)\in E}|N_\Gamma(v,e)|}. \tag{4}$$

This value measures the proportion of false neighbors to all neighbors implied by Γ, with low values corresponding to less ambigous drawings. For $\delta = 1$ the set N^t_Γ contains only the actual neighbors in edge set E, while $\delta \to \infty$ counts all reachable vertices as true neighbors; the remaining false neighbors correspond to vertex pairs in different connected components (i.e. the most ambiguous). Hence for connected graphs, there is some value $\delta_0 \in \mathbb{N}$ for which amb(Γ) drops to zero for all $\delta \geq \delta_0$. Values for $\delta > 1$ correspond to tolerating ambiguous connections if the endpoints are connected by paths of at most δ edges. We use amb$^\delta(\Gamma)$ to denote the ambiguity for a particular value δ.

This ambiguity measure will count all ambiguities due to shallow edge crossings, independent edge ambiguities (Fig. 2(a)), and path endpoint ambiguities (Fig. 4). Thus, Edge-Path bundling will have a non-zero value for this measure. However, in all of the Edge-Path bundling drawings, by definition, there are no independent edge ambiguities.

4.4 Detecting Edge Ambiguity in a Drawing

The approach chosen to detect edge ambiguity is based on the premise that nearby edges with a similar direction may mistakenly have their endpoints interchanged, especially edges that are parallel or cross with a small angle. We thus check if edges are spatially close and if the crossing angle is below the threshold $\theta = 7.5^\circ$, a crossing angle for which empirical evidence [27] shows a strong negative effect on readability.

First, the drawing area is divided into a small, square grid. Two edges are considered ambiguous if an ambiguity is detected in at least one cell. Each grid cell is assigned the intersecting edge segments it contains as well as their respective angle. Internally, curved edges are approximated using polylines, leading to two cases. In the first case, a segment ends in or intersects a grid cell and the angle of the segment is assigned. In the second case, where multiple segments of the polyline intersect the grid cell, the mean of the angles is assigned, under the assumption that the grid cell is small enough such that there is no drastic change of direction.

After this initial assignment, a sliding window processes the cells of the grid. The angles of the assigned segments of an edge are aggregated over all cells in the window which gives us an angle for each edge intersecting the window. Edges are processed in a pairwise manner. Given that two edges intersect the window, the edges must be spatially close. Therefore, the smaller of the two crossing angles is compared to a threshold θ to determine if it is an ambiguity.

Additionally, we need the set of neighbours for the endpoints of an edge $e = (s,t)$ in the sets $N_\Gamma(s,e)$ and $N_\Gamma(t,e)$. When assigning segments to grid cells we iterate over the segments of an edge from source to target and assign an angle in the range $[0, 2\pi]$. This gives enough information to determine the relative position of ambiguous edge segments by comparing their induced angles and deciding if the source vertex of one edge is ambiguous with the source or target vertex of another edge. This is analogously defined for the target vertex.

5 Data

We computed bundled layouts of real and synthetic datasets. The synthetic datasets highlight properties of bundling behaviour. The real world datasets are common in the literature and used for comparison.

Cubes 1R-4R. We created several synthetic datasets that should capture connections between disconnected components, a common feature in real-world datasets. Vertices ($|V| = 100$) are evenly distributed into four components which are subsequently randomly embedded inside axis-parallel cubes with fixed side length s. The cubes are positioned top left, top right, bottom left and bottom right with fixed space of $2s$ between the right edges of the left cubes and the left edges of the right cubes, see Fig. 6. We introduce a distance Δ between the bottom edge of the top cubes and the top edge of the bottom cubes. In Cubes 1R and 4R $\Delta = s/10$, which creates space between the cube pairings. In Cubes 2R $\Delta = 0$ and for Cubes 3R $\Delta = -s/5$ which results in overlapping top and bottom cubes. A random spanning tree is used to connect vertices inside their cube component and additional edges are randomly added. In Cubes 1R-3R the left cubes are connected with their right counterpart and in Cubes 4R the components are connected diagonally. For connecting the components we added $|V|/2$ edges randomly. In the directed variation we additionally specify half of the edges going from left to right and the other half from right to left.

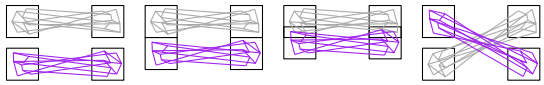


Fig. 6. Illustration of the Cubes 1R–4R datasets.

Noise. This dataset is based on the observation by Lhuillier et. al [36] that bundling approaches do bundle without support in the underlying data. For the Noise dataset, we embedded $|V| = 1000$ vertices randomly in a square and connected vertices to form a perfect matching, resulting in $|E| = |V|/2$ edges and connected components.

US Airlines. The US Airlines dataset has been introduced by [25] and is commonly used in bundling publications. The dataset depicts flight paths from source to target airports in the US and has $|V| = 235$ vertices and $|E| = 2101$ edges. It has one connected component.

Migrations. This dataset [15] shows migration in the US [49] and consists of a set of trails which we converted into a graph. The directed graph has $|V| = 1702$ vertices and $|E| = 9726$ edges. The number of vertices is slightly lower compared to earlier literature (1712) when we converted the trail sets. The undirected graph has $|E| = 6487$ edges. It has 28 connected components.

Air Traffic. This dataset consists of global flights and has $|V| = 1533$ vertices and $|E| = 14825$ edges. It has one connected component.

Amazon Subset. This dataset consists of products with edges indicating that they are commonly co-purchased. We randomly filtered edges from the original graph [33, 34]. It has $|V| = 192k$ vertices and $|E| = 269k$ edges.

6 Experiment Rendering and Bundling Algorithms

To guarantee a fair comparison between the different systems, we rendered all images of the different bundling results in the same way. First, we computed a bundling with the systems' respective implementation. Then, we extracted the bundled edges as polyline approximations of the curves and used this as input for our own rendering. As some systems required a different scaling of the input layout, we proceeded to scale all bundlings to a fixed width of 1600px while keeping the original aspect ratio. In the standard bundling plot, we used a linewidth of 1px to draw the edges and a diameter of 4px for disks representing vertices. We used this plot to calculate the ink reduction of a bundling approach.

To give a better impression of the direction of edges in a bundle, we created a second plot where we applied a perceptually uniform colour map [14] to assign a color to each edge depending on its angle in the straight line drawing. Furthermore, we created plots of the distortion, where we first computed the minimum and maximum distortion over all results of a dataset and applied a sequential colormap [14] to show an overview of where edges are distorted in a bundling. Finally, as we calculate the ambiguity of edges on a per cell basis, we can convert this to an image. We assign each cell the number of intersecting ambiguous edge-pairs and normalize by the maximum over all results of a dataset to achieve a greyscale image that encodes areas of high ambiguity as white spots. These images can be found in the supplementary material.

6.1 Bundling Algorithms in the Experiment

For our experiment, we selected algorithms with available implementations as representatives from major categories of bundling approaches: force-directed bundling, confluent drawings with fixed vertex positions, grid-based approaches, and image-based approaches.

Force-directed Bundling. We used an implementation in D3.js [2] and set the parameters as specified in [25] (spring constant $K = 0.1$, force iterations $I = 60$, subdivision operation $C = 6$, initial subdivision points $P = 1$, increase ratio 2). As the bundling depends on the scale of the input graph, we did not rescale the input layout before bundling.

For **Confluent Drawings.** we used an available implementation [5] based on the results of [10, 57], which we modified to handle graphs with an embedding. This was realized by removing the layout step of the implementation and replacing it with the following. First, we fixed all vertices of the input layout and embedded the routing vertices in the barycenter of neighbouring input vertices. Afterwards, we iteratively moved the routing vertices towards the the barycenter of their neighbours until the layout converged. Generally, confluent bundling algorithms draw and bundle the graph simultaneously, but our approach requires a drawing as input. This means that these approaches are hampered as they cannot modify the layout.

Winding Roads. We used the implementation [31] in Tulip [9].

KDEEB. We adapted an available implementation [4] of KDEEB [29]. A comparison with FFTEB [35] would be preferable, but we were not able to run the implementation. Furthermore, given the computational effort required for our larger experiments we used the GPU implementation which had its parameters fixed by the authors. The layout of the input graph was scaled to the range $[0,1]$.

CUBu We used the available implementation [1] of CUBu [50]. A medium sized kernel was used while most default settings where kept. We increased relaxation to get smoother lines.

Divided Edge Bundling. We used an implementation in Matlab [3] of Selassie *et al.* [45] for directed edge bundling using default parameters.

6.2 Parameter Specification

As discussed in Sec. 3, we can specify several parameters when computing a bundling. We tested different settings and evaluated the resulting ink reduction, distortion, and ambiguity.

We found that a maximum distortion factor $k \in [1.5, 3.0]$ works well for most of the experiments. A distortion factor of less than 1.5 resulted in very little edge bundling. For the Cubes datasets, a distortion factor above 3.0 resulted in overaggregation with ink reduction > 1. In the US Airlines dataset, when increasing the distortion factor more edges were bundled, but increasing above a factor of 3.0 did not produce more bundling, which can be explained by the fact that most shortest paths are below the distortion threshold already. Images of the effect of this parameter on Airlines are contained in the supplementary material. Ultimately, we used $k = 2.0$ for all experiments.

Experiments on the edge weight factor d showed that values of $d \in \{1,2,3\}$ work well. Higher values of d penalise long edges from being included in shortest paths used for bundling. A value of $d = 2$ was chosen for our experiments.

As our algorithm computes a list of control points for each edge, we can use an integer smoothing parameter to pull the rendered curve towards its control points. A smoothing parameter of 1 would use the original list of control points to calculate the curve. A smoothing parameter of 2 would add an additional control point in the middle of the line between every consecutive pair of control points. Increasing the smoothing factor above 2 results in recursively adding more control points thus creating a tighter bundling. In our experiments, we set the smoothing parameter to 2.

7 Experiments

We next discuss our experimental results. KDEEB and CUBu are image-based techniques. As it was easier to adjust the kernel size of CUBu, its result images are shown. High resolution images of all algorithms, including KDEEB, are in the supplementary material.

7.1 Runtimes

We used all the cited implementations specified above. As these implementations are written in a variety of languages with some designed to run on the GPU and others the CPU, direct comparison of runtimes is less informative. However, it was important to note that the image-based techniques, KDEEB and CUBu, have by far the best performance in terms of runtime, processing large graphs in less than a second.

We report the average runtimes of ten executions for Edge-Path bundling. We implemented our bundling algorithm in C++ and ran it on a machine with Ubuntu 20.04 operating system, AMD Ryzen 5 5600x CPU and NVidia RTX 3070: Cubes (1-4)R [128ms], Noise [401ms], Airlines [920ms], Migrations [undirected: 10.6s, directed: 15.3s], Air Traffic [49.6s], and Amazon Subset [7.5 hours].

Furthermore, we also report runtimes for bundling the Air Traffic dataset, the largest dataset presented in an image, with CUBu [5ms incl. rendering], KDEEB [45ms], Force-directed [32.5s], Force-directed (divided) [3.2hrs], Winding Roads [7.6s], and Confluent [157s].

7.2 Synthetic Data

We first discuss the quantitative results on our synthetic data, Cubes as well as the Noise data. By design of the Cubes datasets, there are two disjoint edge sets from left to right, which can potentially be grouped into distinct bundles, but edges from disconnected components should ideally not be mixed. In the Noise data, consisting of disconnected edges, there is nothing to be bundled, topologically speaking. Refer to Table 1 for the scores of the quality metrics and Fig. 7 for the result images. The results for Cubes 1R and 2R were similar to 3R and 4R. Full metric results available in the supplementary material.

Ink Reduction. Naturally, the ink reduction for Straight Line is 1. KDEEB and CUBu consistently and as expected reduce ink the most on all four Cubes instances. Winding Roads is third and then Force-Directed. The stronger ink reduction of the other approaches comes at the cost of overbundling some edges and creating ambiguities. Confluent and Edge-Path have a relatively low reduction; this is expected as the more graph topology-aware bundling algorithms do not create independent edge ambiguities. Results on Noise are similar (see also Fig. 3), where KDEEB and CUBu, Winding Roads, and then Force-Directed reduce ink the most. Confluent and Edge-Path do not change the input layout at all, as they never bundle independent edges.

Distortion. Straight Line, by definition, has a distortion of 1. We report mean and median distortion values. While the mean distortion is generally least for Force-Directed bundling with just a few percent, the median distortion is consistently smallest for Edge-Path, meaning that the majority of edges are unbundled and thus undistorted. Winding Roads is the next best, with CUBu, and KDEEB. Although Confluent has a high mean distortion, its median is 1 indicating it distorts a few edges within the components drastically as the layout cannot be adjusted. The observed distortions in the Noise data is non-existent for Confluent and Edge-Path as there is no bundling. Force-Directed, producing only few and thin bundles, has generally low distortion, followed by Winding Roads, CUBu, and KDEEB.

Ambiguity The ambiguity scores are lowest for Straight Line as it has no bundling with values of 0.47–0.56 caused by shallow-angle edge crossings only. All bundling algorithms increase the ambiguity with KDEEB, CUBu, and Force-Directed having the largest ambiguity scores followed by Winding Roads and Edge-Path. Confluent has the lowest ambiguity score. Algorithms that produce stronger bundling with more ink reduction often produce more independent edge ambiguities. With increasing δ, Edge-Path becomes the best performing algorithm in terms of ambiguity. As expected, on the Noise data, Confluent and Edge-Path again obtain the same initial ambiguity as Straight Line. Force-Directed has the next lowest ambiguity score followed by Winding Roads and KDEEB.

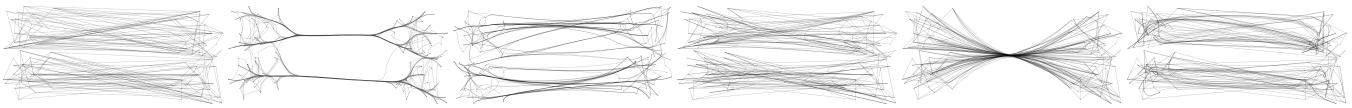
Qualitative. Fig. 7 shows our results. On the Cubes datasets, Force-Directed, KDEEB, CUBu and Winding Roads can mix the two independent streams of edges. Confluent does not mix these streams by design, but has a low degree of bundling. Edge-Path bundling bundles within each stream but does not connect the two disconnected streams. This demonstrates that it is able to avoid grouping unrelated edges together.

7.3 Real Data

Tables 2 and 3 show the quality metrics for the real world data and Fig. 8 through Fig. 11 show the bundled layouts for this data.

7.3.1 Airlines

Ink Reduction. On the (undirected) Airlines dataset, KDEEB achieves the largest ink reduction followed by Winding Roads, CUBu, and Edge-Path. Force-Directed and Confluent achieve the least ink reduction. These scores also match the visual impression of Fig. 8. The ink



(a) Straight Line (b) CUBu (c) Winding Roads (d) Confluent (e) Force-Directed (f) Edge-Path

Fig. 7. The Cubes 2R dataset shows the bundling of two connected components. KDEEB, Winding Roads and Force-Directed bundle edges regardless if edges are connected in the data. Confluent drawings and Edge-Path only bundle edges in their respective components.

Table 1. Scores of the quality metrics for the undirected synthetic datasets and all bundling algorithms. Column dist_Γ gives mean and median. Columns amb^δ are only shown for $1 \leq \delta \leq 5$ if there are non-zero entries. For Noise, all amb^δ are equal as the graph consists of independent edges.

	Cubes 3R							Cubes 4R								Noise			
	ink_J	dist		amb^1	amb^2	amb^3	amb^4	ink_J	dist		amb^1	amb^2	amb^3	amb^4	amb^5	ink_J	dist		amb^δ
Straight-Line	1.00	1.00	1.00	0.50	0.32	0.08	0.01	1.00	1.00	1.00	0.56	0.37	0.12	0.02	0.00	1.00	1.00	1.00	**0.50**
Force	0.79	**1.04**	1.02	0.87	0.61	0.19	0.01	0.70	**1.01**	**1.00**	0.88	0.65	0.30	0.12	0.10	0.98	**1.00**	**1.00**	0.67
Confluent	0.93	1.23	**1.00**	**0.73**	**0.42**	**0.13**	0.01	0.93	1.20	**1.00**	**0.79**	0.54	0.28	0.16	0.14	1.00	**1.00**	**1.00**	**0.50**
WindingR	0.50	**1.04**	1.03	0.81	0.55	0.19	0.01	0.51	1.04	1.02	0.81	0.57	0.25	0.09	0.05	0.64	1.06	1.05	0.96
KDEEB	**0.22**	1.11	1.07	0.90	0.64	0.22	0.01	**0.26**	1.13	1.09	0.94	0.81	0.60	0.48	0.46	**0.43**	1.19	1.18	0.99
CUBu	0.28	1.08	1.07	0.90	0.63	0.20	0.01	0.32	1.10	1.09	0.95	0.81	0.60	0.49	0.47	0.54	1.16	1.15	0.99
Edge-Path	0.84	1.05	**1.00**	0.80	0.49	**0.13**	0.01	0.91	1.04	**1.00**	**0.79**	**0.50**	**0.16**	**0.02**	**0.00**	1.00	**1.00**	**1.00**	**0.50**

reduction result would be expected as Edge-Path bundling provides a good compromise between ambiguity and visual simplification.

Distortion. Force-Directed performs best with Winding Roads and Edge-Path follow in second. For KDEEB, CUBu, and Confluent we observe higher distortions, which is expected due to the strong ink reduction; for Confluent, as stated before, the bundle routing had not been originally designed for pre-embedded graphs.

Ambiguity. The straight-line layout has an ambiguity score of 0.59 due to shallow edge crossings. Force-Directed achieves the least additional ambiguity, followed by Winding Roads and Edge-Path. KDEEB and CUBu are slightly more ambiguous and the poor edge routing in Confluent lets it score highest on ambiguity. This indicates that Confluent, although having good theoretical ambiguity properties, is not well suited if the vertex positions cannot be adjusted. As δ is increased, all approaches perform similarly and go to zero when $\delta = 3$.

Winding Roads and Edge-Path provide similar scores and minimise overaggregation. Winding Roads produces smaller bundles, prone to fewer shallow crossings, but these bundles are not necessarily backed by structural paths. Edge-Path bundling produces fewer larger bundles, all of which are backed by graph structure.

Qualitative. Fig. 8 shows the results for undirected bundling. Images for directed bundling are available in the supplementary material. The undirected drawings of Force-Directed, CUBu and KDEEB can clarify coarse structure in the drawing. In particular, there is a large bundle headed east-west and some of the structure in the densely populated east coast. Winding Roads is able to divide the large bundles into smaller ones but these do not necessarily correspond to graph structure. Edge-Path bundling clarifies the two distinct streams on the west coast follow different paths in the network eastward with Denver seeming to act as a control point. Also, the bundles on the east coast are four distinct path bundles. These path bundles loosely correspond to four airport hubs: Atlanta, Minneapolis, Detroit, and Dallas.

7.3.2 Migrations

Ink Reduction. For the Migrations dataset (Fig. 1), KDEEB achieves the best ink reduction, closely followed by Edge-Path and then Winding Roads. CUBu and then Force-Directed are next and finally Confluent. These results match the visual impression of the bundled layouts and are also reflected in the directed Migrations data.

Distortion. CUBu, Winding Roads, and Edge-Path have the lowest mean distortion scores but very similar medians. Force-Directed and KDEEB follow next. Confluent, as before, scores worst in the distortion metric, which we again attribute to its need to compute a layout simultaneously. These results match our expectations and are similarly observed in the directed Migrations data.

Ambiguity. The Migrations Straight-Line layout has a high baseline ambiguity of 0.7. All methods produce similar ambiguity scores for $\delta = 1$. Thus, at this value of δ, shallow edge crossings dominate and all are equally ambiguous. However, as δ grows, shallow edge crossings matter less in the measure and bundling disconnected edges increase in importance. We show $\delta = 1 \ldots 5$ and almost immediately, Edge-Path bundling outperforms all approaches. It is competitive with Straight-Line as it is able to pull edges connected at small distances away from edges that are not necessarily connected at small values of δ.

On the Migrations dataset Edge-Path bundling is a clear winner with high ink reduction, low distortion, and low ambiguity competitive with straight line drawings.

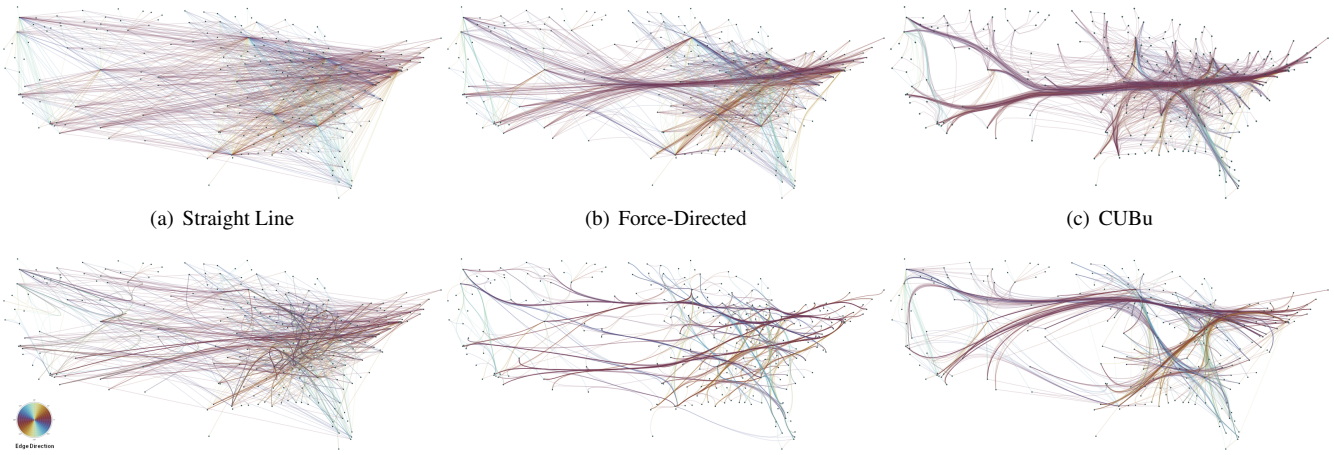
Qualitative. Fig. 1 shows the undirected bundling approaches on Migrations. Force-Directed and CUBu can present the main east-west direction of flow and the graph structure on the east coast to some degree, but aggregate to a very high degree and can include unrelated edges. Winding Roads further subdivides these bundles, but again some of the structure can be obfuscated by clustering unrelated edges together. Edge-Path only shows flows that are in the underlying graph which are not seen in the other drawings. The east-west flows actually correspond to two distinct paths: one heading northward and one heading towards Texas. On the east coast, the flows are further divided into a smaller number of compact streams that reflect paths in the data.

Table 3 shows the metrics for directed Migrations with similar results. Fig. 9 compares directed edge bundling using a force-Directed approach to our own. The force directed approach is able to divide out streams heading in different directions as can be seen in the east-west parallel flows across the country and the north-south flows on the east and west coast. However, these patterns contain a number of unrelated edges. In Edge-Path bundling, the east-west flow divides into three parts around Texas with the thickest bundle heading towards the great lakes region. On the east coast, more detail is revealed in the north-south direction. All of this detail depends on the structure of the underlying graph as unrelated edges are not bundled.

7.3.3 Air Traffic

Ink Reduction. KDEEB performs the best with Winding Roads and Edge-Path in second. There is a large gap between the other approaches and these three algorithms. On the directed datasets, Edge-Path and CUBu outperform force-Directed. Edge-Path bundling aims at bundling well while reducing the amount of ambiguity in the network and therefore it is competitive with top bundling approaches.

Distortion. Edge-Path bundling ranks fourth only outperforming confluent, but when looking at the median distortion it performs the same as all other approaches. Confluent does not perform well on distortion as it is forced to retain the input layout meaning that bicliques could be separated by large distances. Therefore, on distortion, Edge-Path performs similarly to all approaches. For directed datasets, Edge-Path and CUBu outperform Divided.



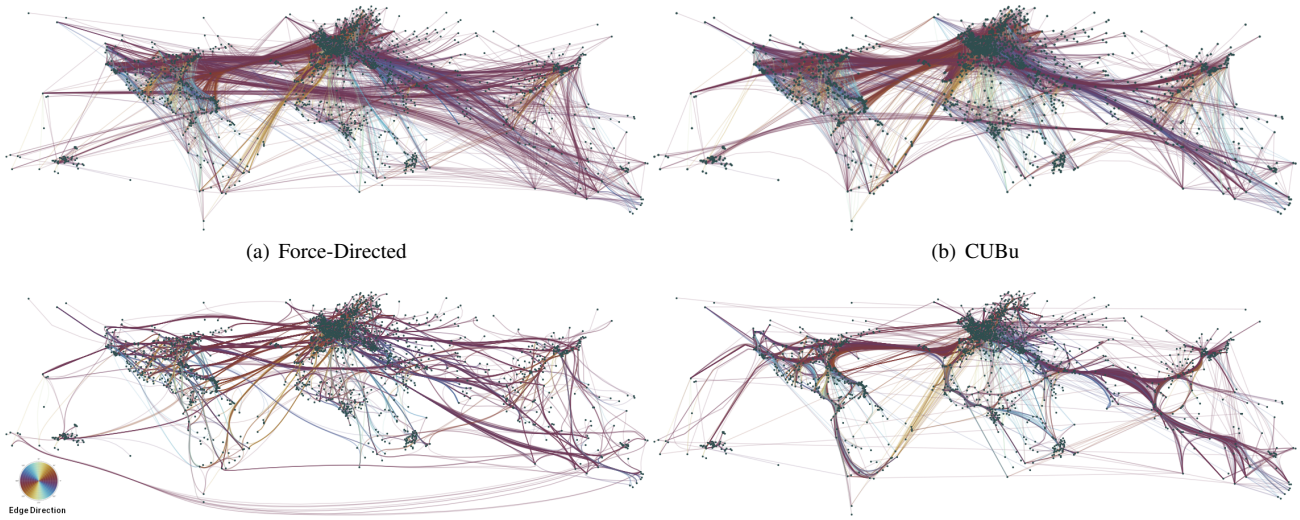
(a) Straight Line (b) Force-Directed (c) CUBu

(d) Confluent (e) Winding Roads (f) Edge-Path

Fig. 8. Airlines (undirected). **(a)** Input drawing. **(b)** Force-Directed bundling is able to cluster edges into the major flows, but some overaggregation prevents details from being visualised. **(c)** CUBu provides a good bundling, but also has overaggregation. **(c)** Confluent drawings can be imposed on the layout, but as the approach cannot layout the graph bicliques can be distantly located, resulting in suboptimal performance. **(e)** Winding Roads divides the flows into many streams, but these streams can be unfaithful to graph structure. **(f)** Edge-Path bundling aggregates edges using weighted paths. The four prominent bundle intersections on the east coast correspond to major airports: Atlanta, Detroit, Minneapolis, and Dallas.

(a) Edge-Path (b) Force-Directed (Divided) (c) CUBu

Fig. 9. Migrations (directed). **(a)** Directed Edge-Path bundling. **(b)** Divided edge-bundling which uses a forced-based approach. **(c)** CUBu. Directed Edge-Path bundling does not bundle unrelated edges together and can reveal more detail on the east coast and in the east-west flow.



(a) Force-Directed (b) CUBu

(c) Winding Roads (d) Edge-Path

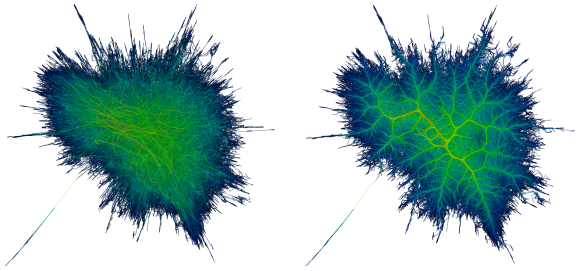
Fig. 10. Results of four algorithms on the Air Traffic network. **(a)** Force-Directed bundling is able to recover the major trajectories but does not strongly bundle the network. **(b)** CUBu strongly bundles the main flows of airtraffic, but can suffer from overaggregation. (c) Winding Roads divides the traffic into many smaller bundles, but this may not be reflective of underlying graph structure. **(d)** With Edge-Path bundling, each bundle necessarily reflects a path in the network. There are separate flows across the Atlantic and Asia that correspond to paths through the network.

Table 2. Scores of the quality metrics for the undirected real-world datasets and all bundling algorithms. Column dist gives mean and median. Columns amb^δ are only shown for $1 \leq \delta \leq 5$ if there are non-zero entries. Bold values highlight the best score in each column.

	US Airlines					Migrations								Air Traffic					
	ink_J	dist		amb^1	amb^2	ink_J	dist		amb^1	amb^2	amb^3	amb^4	amb^5	ink_J	dist		amb^1	amb^2	amb^3
Straight-Line	1.00	1.00	1.00	0.66	0.02	1.00	1.00	1.00	0.71	0.25	0.06	0.02	0.01	1.00	1.00	1.00	0.74	0.06	0.00
Force	0.76	**1.03**	**1.01**	**0.85**	0.03	0.77	1.10	1.06	0.88	0.34	0.10	0.04	0.02	0.79	1.05	1.02	**0.78**	**0.08**	**0.00**
Confluent	0.81	1.29	1.12	0.88	**0.02**	0.89	1.40	1.14	0.90	0.31	0.07	0.03	0.02	0.85	1.47	1.28	0.90	0.09	**0.00**
WindingR	0.46	1.06	1.04	0.87	0.03	0.58	1.06	1.04	**0.87**	0.34	0.10	0.04	0.03	0.57	1.05	1.04	0.81	0.12	0.01
KDEEB	**0.30**	1.21	1.15	0.90	0.09	**0.52**	1.14	1.07	0.92	0.46	0.16	0.07	0.05	**0.39**	1.05	1.02	0.91	0.23	0.02
CUBu	0.61	1.10	1.10	0.89	0.04	0.68	**1.05**	**1.02**	0.90	0.32	0.08	0.03	**0.01**	0.70	**1.00**	**1.00**	0.81	0.13	0.01
Edge-Path	0.56	1.08	1.05	0.87	0.04	0.54	1.07	1.03	0.89	**0.24**	**0.03**	**0.01**	**0.01**	0.58	1.11	1.08	0.89	0.15	0.01

Table 3. Scores of the quality metrics for the directed real-world datasets and the directed bundling algorithms. Column dist gives mean and median. Columns amb^δ are only shown for $1 \leq \delta \leq 5$ if there are non-zero entries. Bold values highlight the best score in each column.

	US Airlines					Migrations								Air Traffic					
	ink_J	dist		amb^1	amb^2	ink_J	dist		amb^1	amb^2	amb^3	amb^4	amb^5	ink_J	dist		amb^1	amb^2	
Straight-Line	1.00	1.00	1.00	0.71	0.02	1.00	1.00	1.00	0.71	0.25	0.06	0.02	0.01	1.00	1.00	1.00	0.74	0.06	
Divided	0.75	1.08	1.04	0.87	0.03	0.79	1.11	1.06	**0.89**	0.34	0.10	0.03	0.02	0.84	1.41	1.39	0.86	0.11	
CUBu	**0.58**	**1.06**	1.04	0.88	0.03	0.62	**1.06**	**1.02**	0.91	0.33	0.08	0.03	0.02	**0.75**	**1.00**	**1.00**	**0.79**	**0.09**	
Edge-Path	0.81	1.07	**1.02**	**0.83**	**0.01**	**0.58**	1.08	1.04	0.90	**0.25**	**0.03**	**0.01**	**0.01**	0.76	1.09	1.05	0.86	0.12	



(a) Edge-Path (b) CUBu

Fig. 11. Edge-Path bundling and CUBu on Amazon Subset. CUBu rendering style is used on both drawings.

Ambiguity. For $\delta = 1$, Force-Directed has the lowest ambiguity with Winding Roads and CUBu in second, followed by Edge-Path. There is less bundling in the force-Directed approach, meaning that it has lower ambiguities. The bundles in Winding Roads are smaller, giving it an advantage in terms of this metric. KDEEB and confluent are not too far away but slightly more ambiguous. With increasing δ, the approaches become comparable rapidly. In the directed case, CUBu and Edge-Path perform similarly with similar findings.

Qualitative. Fig. 10 shows the results of Force-Directed, CUBu, Winding Roads, and Edge-Path bundling on this global airline dataset. Force-Directed is able to simplify the east-west flows across the Atlantic and Pacific, but only at a high level with many of the edges remaining unaggregated. CUBu is able to simplify these flows, but suffers from some overaggregation. Winding Roads divides these flows into several smaller streams, but these streams may not be reflective of graph structure. Edge-Path bundling provides a compromise between level of bundling and ambiguity. In this image, flows from Europe to South America are a separate yellow bundle, indicating few independent paths. Flights directly across the Atlantic are divided into separate bundles that represent paths in the graph as well as detail in the United States being revealed. Flights in Asia split into a triangle of streams towards Australia with north-south flights being in dependant from flights on the continent. Edge-Path bundling is able to reveal structure in this network that is more faithful to the graph structure.

7.3.4 Amazon Subset

Fig. 11 shows the Amazon Subset using the CUBu rendering style. Images of other algorithms that were able to bundle the graph are in the supplementary material. The dataset was too large for our ambiguity metric, so metric values are not reported. CUBu bundling renders a tree-like pattern whereas Edge-Path does not. Edge-Path bundling takes into account the graph structure (faithfulness [39]). CUBu has no such guarantee, because it does not take this structure into account. However, the tree-like structure emerges from bundling edges with similar attributes together such as proximity and direction.

8 Discussion, Limitations, and Conclusion

Edge-Path bundling is not without its ambiguities even though it completely eliminates independent edge ambiguities. As shown in Fig. 4, direct connections between vertices can be ambiguous, but paths between all vertices in the cluster will exist by definition. This edge aggregation is similar to node aggregation strategies [6–8,12,22,51]. In visualisations produced by node aggregation strategies, it is impossible to understand *how* the vertices within the aggregated node connect, only that they are connected; in Edge-Path bundling, we know the bundle is connected and paths exist between all vertices, but specific connections are difficult to read. Interactive techniques, similar to Edgelens [53], could be explored to disambiguate this ambiguity. Edge-Path bundles can be more convoluted as they must follow shortest paths in a graph. Detection of these loops would solve this issue but remains future work.

When bundling directed graphs, Edge-Path bundling can give poor results, since a directed graph may have many forward and backward edges between vertex pairs. This can lead to edges being bundled along the same path in opposite directions. When combined with overplotting, this may result in indistinguishable bundles. Future work could focus on a better routing approach for bundles or avoiding to use the same path in both directions or using a parallel bundling style [50]. Edge-Path bundling takes a graph and a layout as input to compute the bundled drawing. Interesting future work would be to design layout algorithms that maximise the Edge-Path bundling as part of the layout process.

Our implementation of Edge-Path bundling has a worst-case complexity of $O(|E|^2 \log |V|)$, but still works well on graphs of up to ten thousand edges. For dense networks where $|E|$ approaches $|V|^2$, the performance deteriorates. From a theoretical standpoint, finding approximations or algorithms that are faster are of interest. Multilevel bundling [21], faster shortest path implementations [11] or parallelisation [56] could be applied to improve the practical performance. Similarly, the scalability of the ambiguity metric is an open problem.

We presented a method for bundling graphs by considering Edge-Path pairs. The resulting drawings do not incur independent edge ambiguities and are more resilient to creating signal through bundling when there is none. Our resulting bundles provide a new compromise between degree of bundling and faithfulness to the graph structure.

Acknowledgments

We acknowledge funding by the Vienna Science and Technology Fund (WWTF) through project ICT19-035, UKRI EP/V033670/1 and Academy of Finland grant 327352.

REFERENCES

[1] CUBu - Implementation. https://webspace.science.uu.nl/~telea001/InfoVis/CUBu. [Online; accessed 5. Aug. 2021].

[2] d3.ForceBundle - Github. https://github.com/upphiminn/d3.ForceBundle. [Online; accessed 5. Aug. 2021].

[3] Divided Edge Bundling - Implementation. https://github.com/kakearney/divedgebundle-pk. [Online; accessed 5. Aug. 2021].

[4] KDEEB - Implementation. http://recherche.enac.fr/~hurter/KDEEB.html. [Online; accessed 5. Aug. 2021].

[5] pconfluent - Github. https://github.com/jxz12/pconfluent. [Online; accessed 5. Aug. 2021].

[6] J. Abello, F. van Ham, and N. Krishnan. ASK-GraphView: a large scale graph visualization system. *IEEE Trans. Visualization and Computer Graphics*, 12(5):669–676, 2006. doi: 10.1109/TVCG.2006.120

[7] D. Archambault, T. Munzner, and D. Auber. GrouseFlocks: Steerable exploration of graph hierarchy space. *IEEE Trans. Visualization and Computer Graphics*, 14(4):900–913, 2008. doi: 10.1109/TVCG.2008.34

[8] D. Archambault, T. Munzner, and D. Auber. Tugging graphs faster: Efficiently modifying path-preserving hierarchies for browsing paths. *IEEE Trans. Visualization and Computer Graphics*, 17(3):276–289, 2011. doi: 10.1109/TVCG.2010.60

[9] D. Auber, D. Archambault, R. Bourqui, M. Delest, J. Dubois, A. Lambert, P. Mary, M. Mathiaut, G. Melançon, B. Pinaud, B. Renoust, and J. Vallet. Tulip 5. In R. Alhajj and J. Rokne, eds., *Encyclopedia of Social Network Analysis and Mining*, pp. 3185–3212. Springer, 2018. doi: 10.1007/978-1-4939-7131-2_315

[10] B. Bach, N. H. Riche, C. Hurter, K. Marriott, and T. Dwyer. Towards unambiguous edge bundling: Investigating confluent drawings for network visualization. *IEEE Trans. Visualization and Computer Graphics*, 23(1):541–550, 2017. doi: 10.1109/TVCG.2016.2598958

[11] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck. Route planning in transportation networks. In L. Kliemann and P. Sanders, eds., *Algorithm Engineering*, vol. 9220 of *LNCS*, chap. 2, pp. 19–80. Springer, 2016. doi: 10.1007/978-3-319-49487-6_2

[12] V. Batagelj, F. J. Brandenburg, W. Didimo, G. Liotta, P. Palladino, and M. Patrignani. Visual analysis of large graphs using (x,y)-clustering and hybrid visualizations. *IEEE Trans. Visualization and Computer Graphics*, 17(11):1587–1598, 2011. doi: 10.1109/TVCG.2010.265

[13] M. Bruls, K. Huizing, and J. J. van Wijk. Squarified treemaps. In W. C. de Leeuw and R. van Liere, eds., *2nd Joint Eurographics - IEEE TCVG Symposium on Visualization*, pp. 33–42. Eurographics Association, 2000. doi: 10.1007/978-3-7091-6783-0_4

[14] F. Crameri, G. E. Shephard, and P. J. Heron. The misuse of colour in science communication. *Nature communications*, 11(1):1–10, 2020.

[15] W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li. Geometry-based edge clustering for graph visualization. *IEEE Trans. Visualization and Computer Graphics*, 14(6):1277–1284, 2008. doi: 10.1109/TVCG.2008.135

[16] M. Dickerson, D. Eppstein, M. T. Goodrich, and J. Y. Meng. Confluent drawings: Visualizing non-planar diagrams in a planar way. *J. Graph Algorithms and Applications*, 9(1):31–52, 2005. doi: 10.7155/jgaa.00099

[17] D. Eppstein, M. T. Goodrich, and J. Y. Meng. Delta-confluent drawings. In P. Healy and N. Nikolov, eds., *Graph Drawing (GD'05)*, vol. 3843 of *LNCS*, pp. 165–176. Springer, 2006. doi: 10.1007/11618058_16

[18] D. Eppstein, D. Holten, M. Löffler, M. Nöllenburg, B. Speckmann, and K. Verbeek. Strict confluent drawing. *J. Computational Geometry*, 7(1):22–46, 2016. doi: 10.20382/jocg.v7i1a2

[19] O. Ersoy, C. Hurter, F. Paulovich, G. Cantareiro, and A. Telea. Skeleton-based edge bundling for graph visualization. *IEEE Trans. Visualization and Computer Graphics*, 17(12):2364–2373, 2011. doi: 10.1109/TVCG.2011.233

[20] H. Förster, R. Ganian, F. Klute, and M. Nöllenburg. On strict (outer-) confluent graphs. In D. Archambault and C. D. Tóth, eds., *Graph Drawing and Network Visualization (GD'19)*, vol. 11904 of *LNCS*, pp. 147–161. Springer, 2019. doi: 10.1007/978-3-030-35802-0_12

[21] E. R. Gansner, Y. Hu, S. North, and C. Scheidegger. Multilevel agglomerative edge bundling for visualizing large graphs. In *Pacific Visualization Symposium (PacificVis '11)*, pp. 187–194. IEEE, 2011. doi: 10.1109/PACIFICVIS.2011.5742389

[22] S. Golodetz, A. Arnab, I. Voiculescu, and S. Cameron. Simplifying tuggraph using zipping algorithms. *Pattern Recognition*, 103:107257, 2020. doi: 10.1016/j.patcog.2020.107257

[23] M. Hirsch, H. Meijer, and D. Rappaport. Biclique edge cover graphs and confluent drawings. In *Graph Drawing (GD'06)*, vol. 4372 of *LNCS*, pp. 405–416. Springer, 2007. doi: 10.1007/978-3-540-70904-6_39

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

[25] D. Holten and J. J. van Wijk. Force-directed edge bundling for graph visualization. *Computer Graphics Forum*, 28(3):983–990, 2009. doi: 10.1111/j.1467-8659.2009.01450.x

[26] W. Huang, P. Eades, and S.-H. Hong. A graph reading behavior: Geodesic-path tendency. In *Pacific Visualization Symposium (PacificVis'09)*, pp. 137–144. IEEE, 2009. doi: 10.1109/PACIFICVIS.2009.4906848

[27] W. Huang, S.-H. Hong, and P. Eades. Effects of crossing angles. In *Pacific Visualization Symposium (PacificVis'08)*, pp. 41–46. IEEE, 2008. doi: 10.1109/PACIFICVIS.2008.4475457

[28] P. Hui, M. J. Pelsmajer, M. Schaefer, and D. Štefankovič. Train tracks and confluent drawings. *Algorithmica*, 47:465–479, 2007. doi: 10.1007/s00453-006-0165-x

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

[30] B. Johnson and B. Shneiderman. Tree-Maps: A space-filling approach to the visualization of hierarchical information structures. In *Proc. of the 2nd Conference on Visualization '91*, VIS '91, p. 284–291, 1991.

[31] A. Lambert, R. Bourqui, and D. Auber. Winding roads: Routing edges into bundles. *Comput. Graph. Forum*, 29(3):853–862, 2010. doi: 10.1111/j.1467-8659.2009.01700.x

[32] A. Lambert, J. Dubois, and R. Bourqui. Pathway preserving representation of metabolic networks. *Computer Graphics Forum*, 30(3):1021–1030, 2011. doi: 10.1111/j.1467-8659.2011.01951.x

[33] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. In J. Feigenbaum, J. C. Chuang, and D. M. Pennock, eds., *Proceedings 7th ACM Conference on Electronic Commerce (EC-2006)*, pp. 228–237. ACM, 2006. doi: 10.1145/1134707.1134732

[34] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014. [Online; accessed 7. Jul. 2021].

[35] A. Lhuillier, C. Hurter, and A. Telea. FFTEB: edge bundling of huge graphs by the fast fourier transform. In *Pacific Visualization Symposium (PacificVis'17)*, pp. 190–199, 2017. doi: 10.1109/PACIFICVIS.2017.8031594

[36] A. Lhuillier, C. Hurter, and A. Telea. State of the art in edge and trail bundling techniques. *Computer Graphics Forum*, 36(3):619–645, 2017. doi: 10.1111/cgf.13213

[37] S. Luo, C. Liu, B. Chen, and K. Ma. Ambiguity-free edge-bundling for interactive graph visualization. *IEEE Trans. Visualization and Computer Graphics*, 18(5):810–821, 2012. doi: 10.1109/TVCG.2011.104

[38] Q. H. Nguyen, P. Eades, and S. Hong. On the Faithfulness of Graph Visualizations. In *Pacific Visualization Symposium (PacificVis'13)*, pp. 209–216. IEEE, 2013. doi: 10.1109/PacificVis.2013.6596147

[39] Q. H. Nguyen, P. Eades, and S. Hong. Towards faithful graph visualizations. *CoRR*, abs/1701.00921, 2017.

[40] Q. H. Nguyen, S. Hong, and P. Eades. TGI-EB: A new framework for edge bundling integrating topology, geometry and importance. In M. J. van Kreveld and B. Speckmann, eds., *Graph Drawing (GD'11)*, vol. 7034 of *LNCS*, pp. 123–135. Springer, 2011. doi: 10.1007/978-3-642-25878-7_13

[41] G. Palmas, M. Bachynskyi, A. Oulasvirta, H. P. Seidel, and T. Weinkauf. An edge-bundling layout for interactive parallel coordinates. In *Pacific Visualization Symposium (PacificVis'14)*, pp. 57–64. IEEE, 2014. doi: 10.1109/PacificVis.2014.40

[42] D. Phan, L. Xiao, R. B. Yeh, P. Hanrahan, and T. Winograd. Flow map layout. In J. T. Stasko and M. O. Ward, eds., *Information Visualization (InfoVis'05)*, pp. 219–224. IEEE, 2005. doi: 10.1109/INFVIS.2005.1532150

[43] S. Pupyrev, L. Nachmanson, S. Bereg, and A. E. Holroyd. Edge routing with ordered bundles. *Computational Geometry*, 52:18–33, 2016. doi: 10.1016/j.comgeo.2015.10.005

[44] D. Reniers, L. Voinea, O. Ersoy, and A. Telea. The solid* toolset for software visual analytics of program structure and metrics comprehension: From research prototype to product. *Science of Computer Programming*, 79:224–240, 2014. doi: 10.1016/j.scico.2012.05.002

[45] D. Selassie, B. Heller, and J. Heer. Divided edge bundling for directional network data. *IEEE Trans. Visualization and Computer Graphics*,

17(12):2354–2363, 2011. doi: 10.1109/TVCG.2011.190

[46] A. Telea and O. Ersoy. Image-based edge bundles: Simplified visualization of large graphs. *Computer Graphics Forum*, 29(3):843–852, 2010. doi: 10.1111/j.1467-8659.2009.01680.x

[47] N. Toeda, R. Nakazawa, T. Itoh, T. Saito, and D. Archambault. Convergent drawing for mutually connected directed graphs. *J. Visual Languages & Computing*, 43:83–90, 2017. doi: 10.1016/j.jvlc.2017.09.004

[48] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[49] United States Census Bureau. County-to-county migration flows files: Census 2000, 2000. [Online; accessed 7. Jul. 2021].

[50] M. van der Zwan, V. Codreanu, and A. Telea. CUBu: universal real-time bundling for large graphs. *IEEE Trans. Visualization and Computer Graphics*, 22(12):2550–2563, 2016. doi: 10.1109/TVCG.2016.2515611

[51] F. van Ham and J. J. van Wijk. Interactive visualization of small world graphs. In *IEEE Symposium on Information Visualization*, pp. 199–206, 2004. doi: 10.1109/INFVIS.2004.43

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

[53] N. Wong, S. Carpendale, and S. Greenberg. Edgelens: An interactive method for managing edge congestion in graphs. In *Symposium on Information Visualization (InfoVis'03)*, pp. 51–58. IEEE, 2003. doi: 10.1109/INFVIS.2003.1249008

[54] J. Wu, L. Yu, and H. Yu. Texture-based edge bundling: A web-based approach for interactively visualizing large graphs. In *Big Data*, pp. 2501–2508. IEEE, 2015. doi: 10.1109/BigData.2015.7364046

[55] J. Wu, F. Zhu, X. Liu, and H. Yu. An information-theoretic framework for evaluating edge bundling visualization. *Entropy*, 20(9):625, 2018. doi: 10.3390/e20090625

[56] Y. Zhang, X. Liao, L. Gu, H. Jin, K. Hu, H. Liu, and B. He. AsynGraph: Maximizing data parallelism for efficient iterative graph processing on gpus. *ACM Trans. Archit. Code Optim.*, 17(4):29:1–29:21, 2020. doi: 10.1145/3416495

[57] J. X. Zheng, S. Pawar, and D. F. M. Goodman. Further towards unambiguous edge bundling: Investigating power-confluent drawings for network visualization. *IEEE Trans. Visualization and Computer Graphics*, 27(3):2244–2249, 2021. doi: 10.1109/TVCG.2019.2944619