The Visual Computer (2019) 35:739–751
https://doi.org/10.1007/s00371-018-1509-7

ORIGINAL ARTICLE

Visualizing large graphs by layering and bundling graph edges

Zhuang Cai[1] · Kang Zhang[2,3] · Dong-Ni Hu[1]

Published online: 30 April 2018

Abstract
Edge bundling has been widely used to reduce visual clutter and reveal high-level edge patterns for large graphs. Due to strong edge attraction, bundled results often show unnecessary curvature and tangling at bundle intersections. Inappropriate bundling may fail to reveal true data patterns and even mislead users. This paper presents a parameterizable 6-step edge bundling approach called LEB that reveals the patterns of the input graph, with distinguishable and traceable bundles. The bundling results by LEB are also adjustable by tuning a small number of parameters. We have conducted a user experiment to test and compare LEB with previous approaches. The experiment on three datasets (including two common ones) demonstrates LEB's superiority over previous approaches in visualizing data patterns. Our implementation with reusable computation also delivers an execution speed fast enough for real-time interaction and animation.

Keywords Graph visualization · Visual clutter · Edge bundling · Edge routing · Layered approach · Evaluation

1 Introduction

Node-link diagrams for visualizing graphs have been widely used to represent relational datasets in many application domains, such as network analysis and geographical data visualization [20]. With increasing scale of graphs, however, traditional straight-link diagrams lose their readability due to visual clutter. Visual clutter, caused by congestion, edge-crossing and node overlapping, leads to the difficulty or impossibility of revealing the high-level structure (i.e., skeleton or backbone).

Edge clutter reduction techniques [8,19,25,36] can be classified into three categories: appearance, spatial distortion and animation [4]. This paper focuses on the spatial distortion techniques, in particular, edge bundling techniques.

✉ Kang Zhang
kzhang@utdallas.edu

Zhuang Cai
caizhuang@tju.edu.cn

Dong-Ni Hu
dongnihu@tju.edu.cn

1 School of Computer Software, Tianjin University, Tianjin, China

2 Department of Computer Science, The University of Texas at Dallas, Richardson, USA

3 Faculty of Information Technology, Macau University of Science and Technology, Macau, China

Edge bundling draws edges by curves instead of straight lines in general node-link diagrams, where nodes have fixed positions in most typical applications. In some applications, nodes' positions have semantic meanings, such as latitude and longitude in most flow maps and network routes. In other applications where nodes are movable, a large number of edges could generate substantial visual clutter, failing to display any data patterns. Edge bundling groups spatially approximate edges into so-called *bundles*, and each bundle could be considered an independent visual object rather than a combination of many edges. This characteristic differs edge bundling from other clustering abstractions, which depends on data attributes. In other words, edge bundling focuses on distorting and/or repositioning graph edges to achieve visual clustering.

Several edge bundling algorithms and methods have been proposed. They follow different paradigms and principles. For example, hierarchical edge bundling (i.e., HEB [13]) is proposed for compound graphs, which are comprised of a tree and an adjacency graph. Force-directed edge bundling (i.e., FDEB [14]) uses a self-organized physical system, while winding roads (i.e., WR [17]) relies on the shortest path algorithm. Other principles, such as constrained optimization and image processing techniques, also are introduced to bundle edges.

The results rendered by existing approaches often show strong edge attraction at the intersections of bundles. Take for example in Fig. 1a, though similar edges are properly

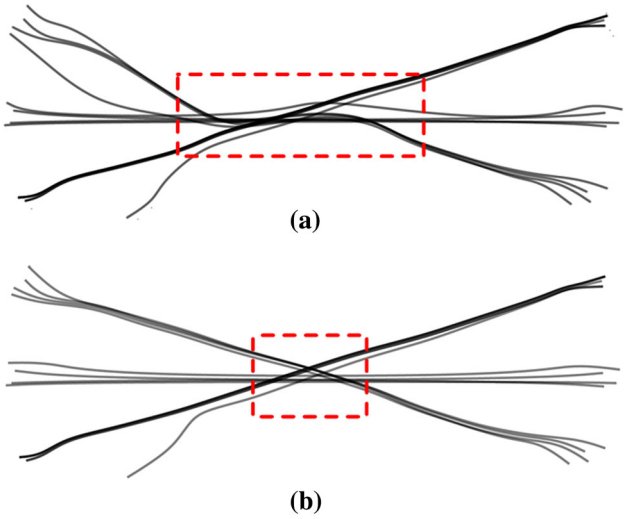


Fig. 1 A motivating example. Top: tangling intersection; bottom: no tangling by our approach

aggregated into bundles at their ends, the bundles tangle with each other in their intersection region. This type of tangling would cause the viewer to lose visual cues, e.g., bending trend, thickness, and is, therefore, the major obstacle for the distinguishability and traceability of bundles. A more distinguishable and traceable solution should retain bundles' independence. In Fig. 1b using our approach, the three bundles have no interference at the intersection and one can identify a bundle's both ends. We therefore consider Fig. 1b to be more traceable than Fig. 1a.

This paper presents a new edge bundling approach, called *LEB*, based on the concept of layers to minimize the interferences between different bundles (as demonstrated in Fig. 2). LEB first temporarily removes short edges and then discovers several primary directions for all the remaining long edges in the given graph. It then discretizes the display space into a regular but low resolution grid and considers the grid as formed by multiple layers, each representing one primary direction. Each edge is routed on the corresponding direction layer using the least-cost property of the A* algorithm [10]. A relaxation scheme is then used to smooth zigzag edges. Edge bundles are finally distinguished with difference colors, based on the bundle separation and bundling strength information, which show the graph's backbone.

The paper is organized as the following. Section 2 reviews previous work on edge bundling. Section 3 presents the 6-step pipeline of LEB with detailed description of each step, followed by Sect. 4 on the description of seven parameters. Section 5 presents the case studies on three datasets, including two used by most previous approaches. Section 6 reports the implementation details and performance statistics on the three datasets. Section 7 discusses our evaluation results on LEB, and Sect. 8 concludes the paper.

2 Related work

Edge bundling (edge aggregation, edge clustering) has attracted an increasing interest not only for graph visualization [23,24,28,40], but also for flow map visualization

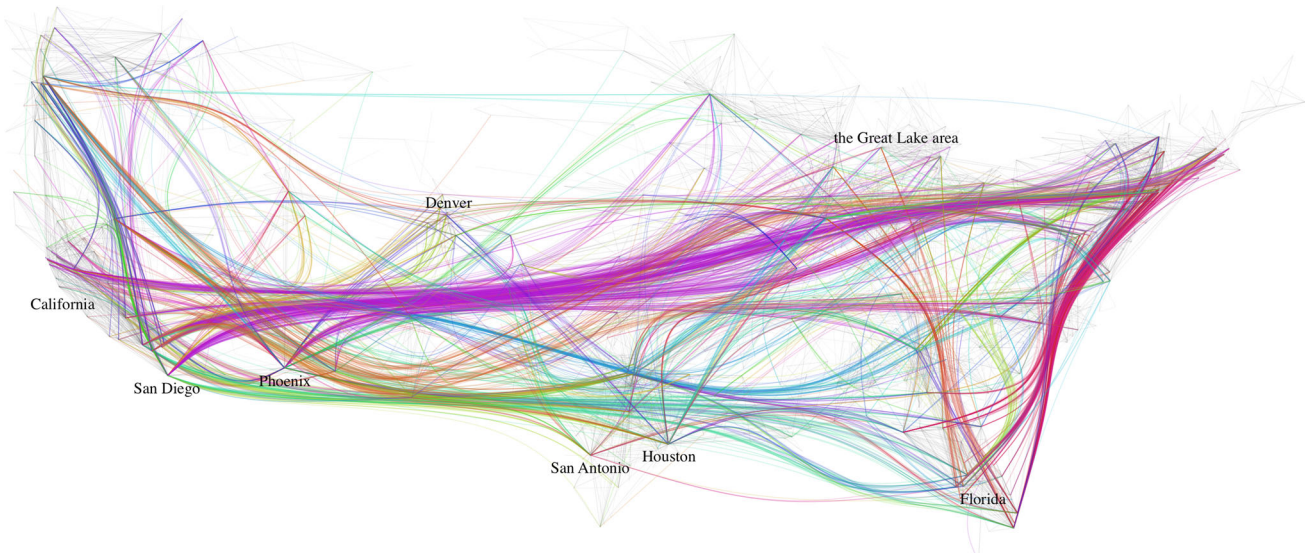


Fig. 2 Edge bundling using our layered approach with colors showing clear patterns of US migrations dataset: a large purple bundle between south–west and north–east represents a large number of migration movements between California and the Northeastern region. Two bundles on the right indicate the movements of migrants between the Great Lake area and Florida and between Northeast and Florida. A small blue bundle between Denver and Houston can be traced even if it is crossed by many larger bundles. Several other migration destinations can also be identified, such as San Antonio, Phoenix and San Diego

and parallel coordinate visualization [22,26,35,39]. Despite different underlying data types, edge bundling aims at grouping similar edges, which encode relations, into bundles to show the potential high-level patterns. Zhou et al. [37] review and classify edge bundling algorithms into three categories according to the underlying paradigms. This paper inherits this classification, but modifies it.

The first category is the so-called *geometry-based* techniques, consisting of hierarchal edge bundles (HEB) [13], geometry-based edge bundling (GBEB) [1] and ambiguity-free edge bundling (AFEB) [21,27]. The main idea of geometry-based approaches is to set appropriate control points to curve edges. Each control point attracts a number of edges close to it, and these aggregated edges form a desired bundle. Choosing an appropriate algorithm and data structure is therefore crucial. HEB handles compound (hierarchy-and-association) graphs. It bends each adjacent edge toward the hierarchical tree path using the B-Spline curve algorithm. Cui et al. propose an edge bundling pipeline for the graphs where node positions are fixed [1]. They use a uniform grid to discretize the display space into cells, which are merged by similarity, and then generate a control mesh by Delaunay triangulation. The intersections of the mesh edges and graph edges are set as control points. To reduce edge ambiguity that occurs when edges are too close to one or more unrelated nodes, Luo et al. [21] present a quadtree-based approach, called AFEB. The quadtree partition is used to detect and disperse edges from unrelated nodes.

In general, geometry-based methods are intuitive and efficient. They, however, usually rely on special scenarios. HEB needs additional hierarchical structures; the space for dispersion desired by AFEB is difficult to find when edge density is excessively high and when bundles cross each other. GBEB often generates bundles that show considerable variations in curvature along the overall bundle directions.

Force-directed edge bundling (FDEB) [14], divided edge bundling (DEB) [29], multilevel agglomerative edge bundling (MINGLE) [9,38] are included in the second category, i.e., *cost-based* approaches. FDEB proposed by Holten and van Wijk [14] uses a self-organizing physical system, in which edges are modeled as flexible springs and electrostatic forces exist between pairs of edges. A combination of compatibility measures, e.g., angle compatibility, position compatibility and visibility compatibility, is used to determine the appropriate magnitude of force and prevent excessive bundling, i.e., over-bundling. Selassie et al. [29] inherit and extend FDEB to directed graphs, thus called DEB. DEB takes connectivity compatibility and edge weight into consideration and help users perceive asymmetries. MINGLE aims at minimizing ink needed to represent curves [9]. It checks whether bundling each edge with its neighbors can share some path to save ink until no more possible ink saving. Due to the time complexity of $O(n^2)$, FDEB and DEB are not scalable to handle large graphs.

The third category is *image-based* techniques, including kernel density estimation graph bundling (KDEEB) [15] and skeleton-based edge bundling (SBEB) [5,33]. Both works adapt image processing techniques to edge bundling and take the image rendered from the straight-line node-link diagram as input. Most of the transformations are performed in the image space. KDEEB employs kernel density estimation (KDE) to estimate the local density of edges and generate density map [15]. Then, edges move to where the local density is maximal. By repeating these steps, edges get closer and eventually converge together. While SBEB uses an existing skeleton-construction algorithm that has been widely used in image processing, instead of relatively simple KDE, to attract edges toward the centerlines of level sets of their distance fields iteratively [5]. Image-based techniques are easy to be accelerated by hardware, such as GPU. Various image-enhancement tools, e.g., density-saturation, shadow and halos effect, could also help users distinguish overlapping bundles. The rendered results of image-based techniques often present a trunk-branch-like feature, which, unfortunately, does not reflect the true characteristics of the data.

Lambert et al. [17] propose a grid-based bundling approach, called *Winding Roads* (WR) that exploits a hybrid grid based on both quadtree and Voronoi diagrams to compute edge rerouting. The grid used by WR plays a similar role as the density map in KDEEB, while WR routes each edge along its shortest path on the grid using Dijkstra shortest path algorithm. Though both WR and our approaches use grid partitioning followed by routing, our grid is of low resolution and layered, and routing is done by an adapted A* algorithm rather than Dijkstra shortest path algorithm. Our approach is therefore able to minimize the interference between dissimilar edges and generate bundles with less curvature and high traceability.

3 Approach overview

To create highly traceable edge bundles that reveal the inherent high-level skeleton (backbone) of the data, we propose a layered approach to edge bundling in this section, and will call our approach LEB. Figure 3 illustrates the LEB pipeline, consisting of six major steps (marked by red circles) as the following:

(1) Discovering primary directions (Sect. 3.1).
(2) Space partitioning with direction layers (Sect. 3.2).
(3) Edge routing (Sect. 3.3).
(4) Edge smoothing with control points (Sect. 3.4).
(5) Identifying and coloring bundles (Sect. 3.5).
(6) Handling short edges (Sect. 3.6).

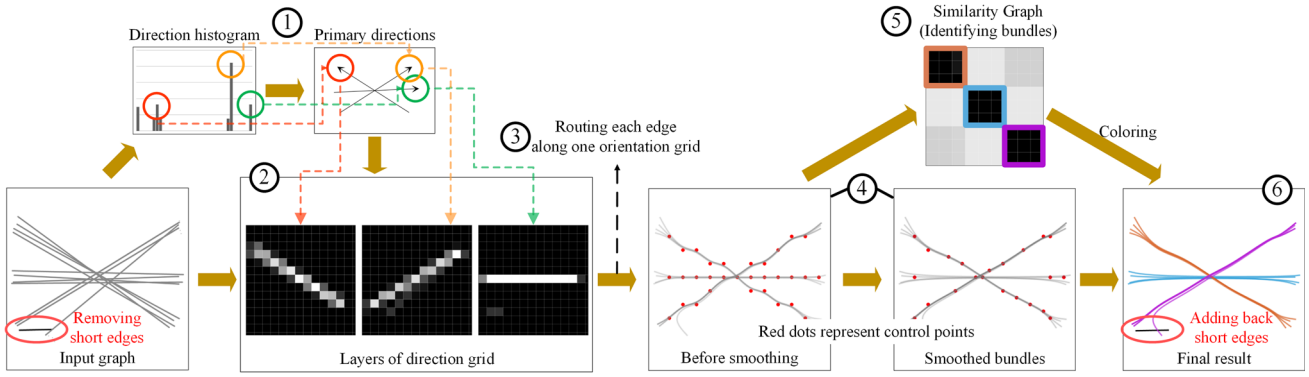


Fig. 3 The pipeline of LEB with six steps, labeled by red circles

We start with an unbundled graph $G = (V, E)$ with a set of nodes V and a set of edges E. The graph is undirected, although our algorithm could be extended to treat directed graphs. First, all short edges under a predefined length threshold are temporarily removed to avoid their interference with the decisions in the subsequent steps (Step 0). The distribution of edge directions is represented as a histogram from which a small number of representative directions, called *primary directions*, are selected (Step 1). Then, a multi-layer low-resolution regular grid is made by decomposing the display space into equal-sized cells and (conceptually) into a number of layers, each representing one primary direction (Step 2). The routing step inputs each primary direction layer and routes all the edges along the primary direction (without influence of other primary directions) using the A* routing algorithm (Step 3). Edge smoothing is then performed to avoid undesired zigzag and thus to enhance traceability (Step 4). Step 5 identifies individual bundles by building a similarity graph for all the edges and colors the bundles to enhance the visual discrimination between them. Finally, Step 6 adds back the temporarily removed short edges before all the edges are rendered.

3.1 Discovering primary directions

Aggregating edges with disparate directions not only leads to tangled bundles, but also winds edges' paths too chaotic to be traceable. We thus decide to set a small number of representative directions in which all the edges are bundled. Such a small number, say N, depends on the distribution of edge directions and thus application-dependent. The edge directions are granulated at a fixed angle called *bin width* that is discussed in Sect. 4. In practice, edge distribution measured by merely the number of edges in each direction does not help effective bundling due to the interference of many short edges. Naturally, long edges should contribute more to the distribution than short ones. We therefore set each edge's length as its weight and compute the weighted

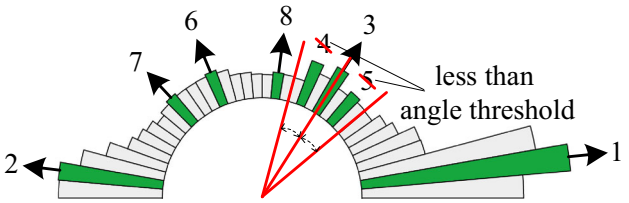


Fig. 4 Direction histogram (0°–180° for any undirected graph), with 8 local maxima (green bars). 1, 2, 3, 6, 7, 8 are chosen as primary directions, after removing 4 and 5 (below angle threshold)

distribution for all the edges to generate a histogram. Figure 4 shows that several peaks (green bars), i.e., local maxima, in the histogram correspond to the directions of those edges that make up the graph's backbone. We first sort all the local maxima in a descending order (1–8 in the example in Fig. 4) and then eliminate the ones whose directions are too "close" to and also lower than their neighboring local maxima (4, 5 in the example). The "closeness" here is defined by the angle formed by the directions of the two neighboring local maxima, which we will call *angle threshold*. We finally select the first N remaining local maxima and use their directions as N primary directions for all the edges to be bundled to.

3.2 Space partitioning with direction layers

Existing edge bundling techniques use several space decomposition approaches to reduce the algorithm complexity, e.g., quadtree partitioning, Voronoi diagram and regular grid. For both efficiency and simplicity, we partition the visualization space into a regular grid of equal-sized square cells. The grid resolution is parameterized and controlled by the user, to be further discussed in Sect. 4. A high-resolution grid with tiny cells would contain too little information to guide edge routing and also generate many local bends. Low resolution and large cells would allow edges to be bundled together and smoother. Through experiment, we decide the cell size of 90×90 pixels for the 1920×1680 visualization space.

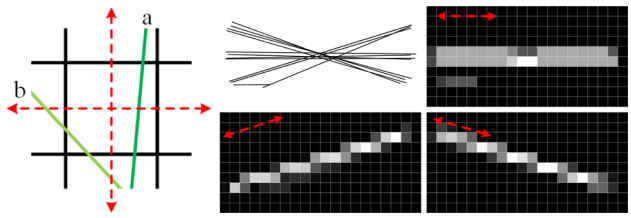


Fig. 5 A cell weighted on edges' lengths within it (left), and three primary directions (layers) with cells of different weights (right)

We also consider the partitioned grid conceptually formed of multiple layers, each representing a primary direction, and thus called a *direction layer*. Separating primary directions into individual layers helps in independent computation in each direction without interference from other directions in the subsequent steps. Figure 1 shows the difference in bundling between separating directions (layered) and not separating directions (conceptually in a single layer). The top figure is generated when a shortest path algorithm is used to route edges without layering, with tangled and unnecessarily curved edges. The bottom figure is generated using the A* routing algorithm, which will be discussed in the next subsection, on individual direction layers separately and then merged. It shows straighter and non-interfering bundles.

We next compute the weight for each cell within every direction layer, independent of other layers. In the example shown in Fig. 5, two edges, a and b, pass through a cell in different directions. Edge a crosses the entire cell, while edge b simply cuts through the cell corner. The segment of a inside the cell is obviously much longer than that of b. Their contributions to the two orthogonal directions (marked as a red cross in Fig. 5) are also different. Edge b contributes nearly equally but insignificantly to both directions, while a contributes almost 100% toward the vertical direction, much more heavily than b. Based on this analysis, we define the weight of a cell at (i, j) as the sum of all passing edges' lengths in the kth $(1 \leq k \leq N)$ primary direction D_k as the following:

$$w_{i,j}^{k} = \sum_{e \in E} \text{length}_{i,j}^{e} \times \delta(d_e, D_k) \tag{1}$$

where $\text{length}_{i,j}^{e}$ denotes the length of edge e inside cell (i, j) and $\delta(d_e, D_k)$ is an indicator function, which returns 1 when edge e's direction d_e is considered part of the primary direction D_k, otherwise 0. We have also tested the continuous cosine distance function instead of the binary indicator function, but the effect is worse. Finally, we deploy a reversed linear scale to map $w_{i,j}^{k}$ to a regular range, i.e., [1, 0], in the implementation. For any given x, the reversed linear scale is defined by:

$$\text{scale}(x, \max, \min) = 1 - \frac{x - \min}{\max - \min} \tag{2}$$

where min and max are the minimal and maximal weights mapped to 1 and 0, respectively.

This process implicitly reinforces the attraction among the edges in similar directions. By this stage, we have constructed the grid of N layers, corresponding to N primary directions. Each cell contains a weight vector sized N. Figure 5 (right) illustrates the grid of three direction layers, in which gray scales represent different weight values, white for the lowest and black for the highest. By contracting all edges to their closest primary directions, we have effectively grouped the edges into bundles.

3.3 Edge routing

The next step is to route edges in the corresponding direction layers obtained in the previous step. We adopt a flexible routing approach to route similar edges along the same path. We adapt the A* routing algorithm, which is designed to find paths and graph traversal [10].

A* algorithm can flexibly control the output paths using a few parameters. Its knowledge-plus-heuristic cost function is defined by

$$h(x) = f(x) + g(x) \tag{3}$$

where $f(x)$ is the past path-cost function and $g(x)$ the future path-cost function. The past path-cost function is the known distance from the initial node to the current node and the future path-cost function is a heuristic estimate of the distance from the current node to the goal. Using A* algorithm with its good performance and well-designed path-cost function $g(x)$, we can control each bundle's bending degree and frequency. We can also set the priority for bundles to pass through unused space.

In general, $g(x)$ should be an admissible heuristic, i.e., it never overestimates the distance to the goal [2,16]. Although a non-admissible $g(x)$ overestimates the distance and derives a 'wrong' path having higher cost, bundles would have less curvature and higher possibility to pass through unused space. An extreme case is when $g(x)$ is much greater than the actual distance, e.g., an positive infinity, we get a completely straight path. Taking advantage of this property, we define $g(x)$ as:

$$g(i) = (|x_i - x_{\text{goal}}| + |y_i - y_{\text{goal}}|) \times k, k \in [0, +\infty) \tag{4}$$

where $|x_i - x_{\text{goal}}| + |y_i - y_{\text{goal}}|$ is the so-called *Manhattan Distance* between the current cell i and the goal cell, and k is a weighting factor adjustable by the user. By changing the overestimation degree of $g(x)$, i.e., increasing or decreasing

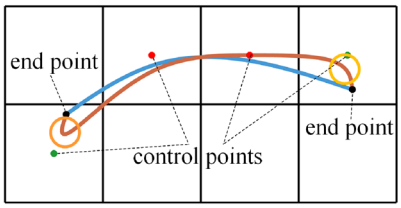


Fig. 6 Removing two control points (green dots) at both ends of an edge to avoid sharp bends

k, we can control the shape of the routed paths for bundles to be rendered on.

Finally, direction layers generated in the previous step help to optimize edge routing. The grid has N layers corresponding to N primary directions. Each edge is routed onto one primary direction that is closest to the edge's direction. Thus, the edges in similar directions share one primary direction.

Due to the inherent property of least-cost routing in A*, our edge routing scheme also guarantees that no pairs of bundles routed in the same direction would intersect each other twice or more (an undesirable effect). This is because, if two or more intersections exist, there would be two or more least-cost paths, which is impossible by the A* algorithm.

3.4 Edge smoothing with control points

Recall that our grid has a low resolution with reasonably large cells. This subsection introduces the preparation step for curve drawing using the B-Spline algorithm, i.e., planting control points inside grid cells. This step aims at minimizing zigzag and winding of edges, that is of crucial importance for smooth bundling.

We develop a 3-step scheme for planting control points as the following. For each edge's path passing through a number of cells, all the control points are set at the centers of these cells. We first remove the control points at both ends that share the cells with the edge's endpoints. This is because the distance between the pair of points inside one cell is short enough to possibly cause a sharp bend, as shown in Fig. 6. We then remove redundant collinear control points along a line to save computation time in rendering B-Spline curves and call this process as *collinear control points removal*, illustrated by the top two figures of Fig. 7.

The final action is to smooth any w-shaped waves. Figure 8a and b illustrate, starting from s, only two types of moves exist, i.e., diagonal and vertical or horizontal move. For a diagonal move, there are eight next possible moves, denoted as $t \rightarrow \{a-h\}$, but cases of $s \rightarrow t \rightarrow \{a, g, h\}$ in Fig. 8a are rejected by our least-cost routing property due to their high costs. Excluding $s \rightarrow t \rightarrow d$ due to collinear control points removal, only $s \rightarrow t \rightarrow \{b, c, e, f\}$ remains as in Fig. 8c. Similarly, for a vertical or horizontal move in Fig. 8b, $s \rightarrow t \rightarrow \{a, b, d, e\}$ remains as in Fig. 8d. By symmetry and

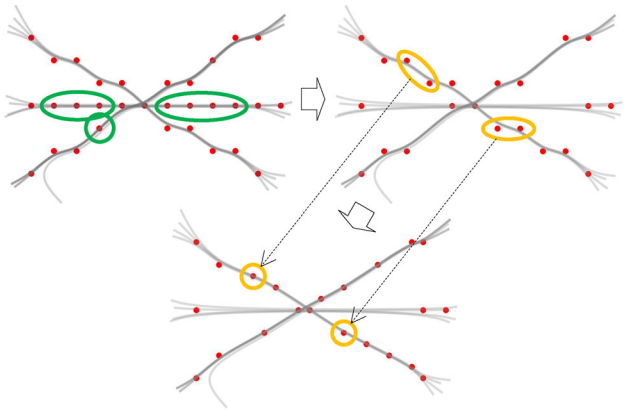


Fig. 7 Removing collinear control points to save computation (horizontal arrow) and smoothing waves (downward arrow)

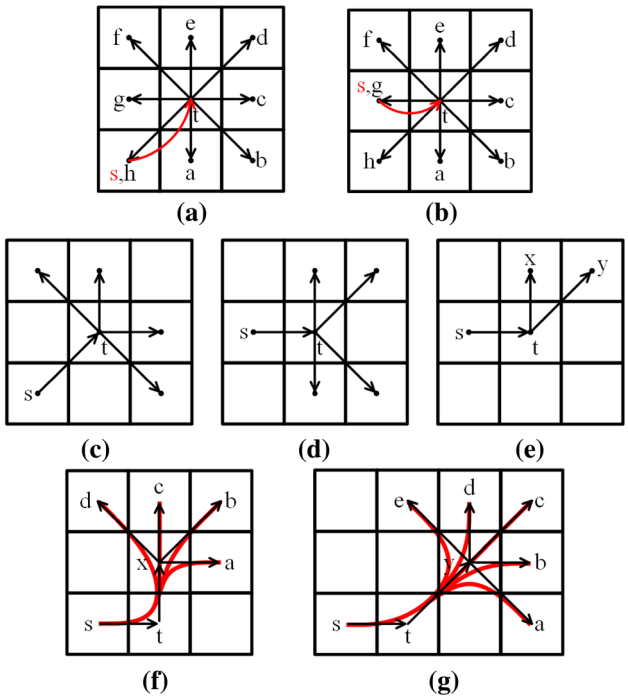


Fig. 8 Various cases for consideration of removing waves

similarity, all the sixteen cases in Fig. 8c and d can be reduced to two possible cases, now denoted as $s \rightarrow t \rightarrow \{x, y\}$ in Fig. 8e.

For $s \rightarrow t \rightarrow x$ in Fig. 8f, four possible cases are:

- $s \rightarrow t \rightarrow x \rightarrow \{c, d\}$, not forming a wave;
- $s \rightarrow t \rightarrow x \rightarrow \{a, b\}$, forming a local wave.

For $s \rightarrow t \rightarrow y$ in Fig. 8g, five possible cases are:

- $s \rightarrow t \rightarrow y \rightarrow \{c, d, e\}$, not forming a wave;
- $s \rightarrow t \rightarrow y \rightarrow \{a, b\}$, forming a wave.

To summarize, our algorithm handles four cases, i.e., $s \to t \to x \to \{a, b\}$ and $s \to t \to y \to \{a, b\}$. It then replaces x and a (or b) by their middle point and replaces y and a (or b) by their middle point. The last step in Fig. 7 shows the resulting effect, where red dots represent the final control points.

3.5 Identifying and coloring bundles

To distinguish different edge bundles, we identify bundles and assign them with different colors. The basic idea is to route similar edges along the same paths and edges sharing longer paths are more likely to be included in the same bundle. Taking advantage of the bundled results through layering as show in Fig. 7 (bottom) and also in line with the previous steps, we use a simple similarity measurement based on the proportion of shared edge lengths, as the following:

$$\text{similarity}_{AB} = \frac{|A \cap B|}{|A|}, \text{ and similarity}_{BA} = \frac{|A \cap B|}{|B|} \quad (5)$$

where $|A \cap B|$ is the length shared by edges A and B, and $|A|$ denotes A's length and $|B|$ denotes B's length. The length here is measured by the number of cells an edge passes through in the corresponding direction layer. Note that the similarity measurement is asymmetric in our definition, implying that edge A is similar to edge B, while B is not necessarily similar to edge A. This happens when the proportion of shared edge length of A is significantly larger than that of B.

By calculating the similarity between every pair of edges, we obtain a *similarity graph*, where nodes are the edges and links are the measurements. First, we use a user-defined threshold t to divide the edges of the similarity graph into two types, connected and unconnected. Using t, one can control the number of identified bundles that indirectly determines the number of colors, which should be no more than the maximal distinguishable categories [12]. Then, we use the standard Tarjan's strongly connected components algorithm to find the strongly connected nodes in the similarity graph [11,32]. These nodes correspond to the edge bundles.

Next, we color these bundles to maximize their visual discrimination. Obviously, the edges within the same bundle should be colored the same. The bundles that do not share any cell are considered to be independent, and assigned with any colors. Those sharing a number of cells are considered interfering each other and assigned with contrasting colors. We use an approach similar to the tree coloring [34] to find a most distant color on the color wheel. For example, for #F00 (0°, red) and #0F0 (120°, green), the output should be #00F (240°, blue).

3.6 Handling short edges

In practice, edge lengths usually vary greatly and many edges are short. Short edges do not contribute significantly to the overall data patterns and also disturb the eventual bundling effects. We therefore filter them out before the space partitioning step, but render them as straight edges in the final rendering step.

4 Rendering and parameter setting

This section presents the rendering scheme and discusses all the parameters introduced above. The small number of parameters is adjustable by the user to obtain the most satisfactory bundling result. In fact, LEB is insensitive to parameter settings. Except s and k as described below, the parameters provide local control without much influence on the overall results.

After being routed, an edge can be seen as a series of points, including two endpoints and a number of control points. We use the B-Spline curve algorithm to plot each edge. B-Spline also allows controllable tension parameters (as coefficients) to be used to straighten or bend the curve and supports local control. LEB supports user-defined opacity so that the edges rendered latter would not obscure those rendered earlier.

LEB uses seven parameters covering various bundling scenarios that are empirically decided.

(a) Number of primary directions N (Sect. 3.1) As a minimal number of representative edge directions for each given graph, N depends on the dataset and generally cannot be pre-computed. An overestimated value may result in under-bundling, i.e., edges of the same direction are not bundled together, whereas an underestimated value leads to serious edge aggregation and low traceability. We initialize N to an overestimated value, say 6, appropriate for most datasets, adjustable by the user.

(b) Bin width of distribution histogram α (Sect. 3.1) α specifies the sampling resolution for edges' direction distribution. It is in fact an angle determining the bin width of the histogram. Obviously, the larger the bin is, the less details remain; the smaller, the more noise. Many researchers have proposed how to determine an optimal bin size [3,7,30,31], but these methods generally rely on strong assumptions on the shape of the distribution. We initialize $\alpha = \frac{\pi}{\lceil\sqrt{M}\rceil}$, where M is the number of data points. Excel histograms and many others also use this formula [6].

(c) Angle threshold for avoiding similar directions β (Sect. 3.1) To generate N primary directions, we introduce β and $\alpha < \beta < \frac{\pi}{N}$. The lower bound, α, is simply the bin width, and the upper bound is $\frac{\pi}{N}$ so that the number of generated primary directions is sufficient to represent edges'

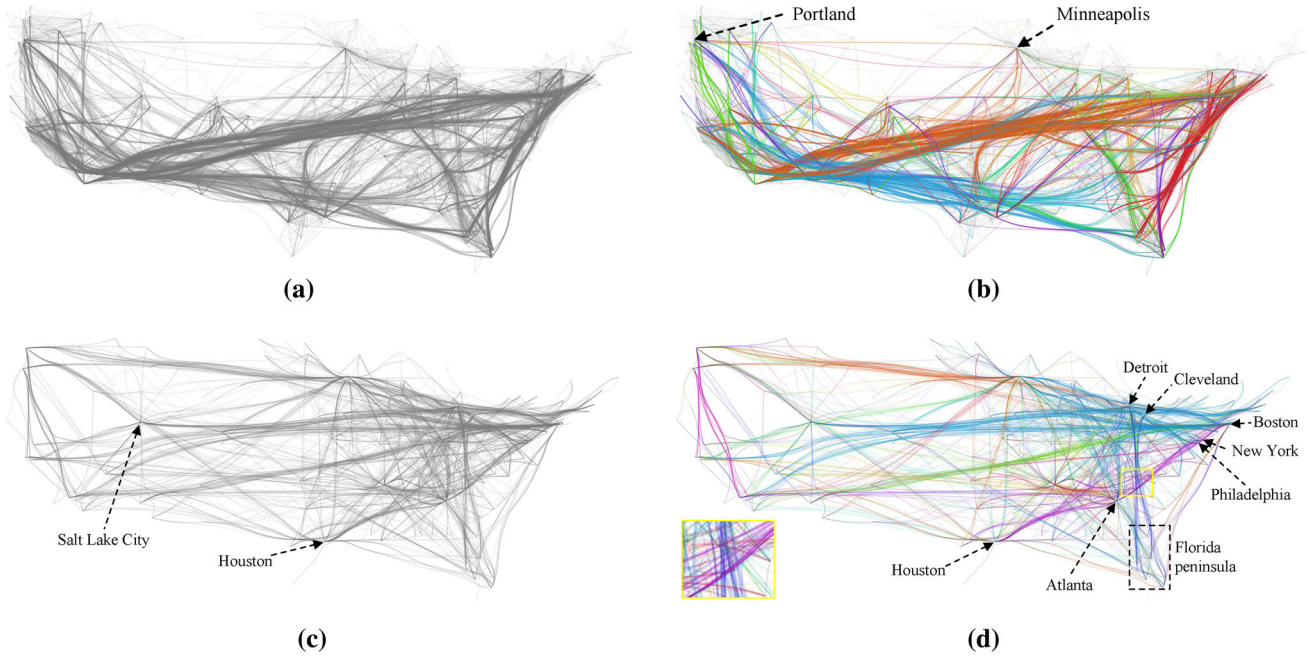


Fig. 9 Bundling results for the US migrations dataset (**a**) and (**b**), and US airlines dataset (**c**) and (**d**)

direction distribution. We initialize β to the midpoint of the two bounds, i.e., $\beta = (\alpha + \frac{\pi}{N})/2$.

(d) Cell size s (Sect. 3.2) The cell size of the grid determines the grid resolution, significantly impacting on the computational complexity. The larger the cell is, the more edges would be aggregated into it, and vice versa. We initialize $s = (W + H)/40$ based on our experiment, where W and H are the width and height of the visualization space.

(e) Estimation factor k (Sect. 3.3) k specifies how much $g(x)$ is overestimated. With an increasing k, bundled lines become straighter. When $k < 1$, implying no overestimation, edges are bundled too tight to be traceable. The values $k \in [1, 3]$ make a good trade-off between under- and over-bundling, and we initialize $k = 2$, based on our experiment.

(f) Similarity threshold t (Sect. 3.5) This threshold is used to determine whether an edge is similar to other edges, i.e., linked to other edges in the similarity graph. It therefore implicitly determines the number of bundles. Different t values may reveal different data characteristics on different levels of details. Furthermore, t merely determines how to color edges and does not contribute to the shape of bundles. In our definition, similarity measurement is between 0 and 1, thus $0 \leq t \leq 1$. We give t a lower bound, that is $0.6 \leq t$ based on our experiment. This enables an appropriate number of bundles to be identified. The number of colors determined would be less than the maximal number of colors for maximal distinguishable categories [12].

(g) Short edge threshold l (Sect. 3.6) l determines the length under which an edge is considered "short". We use the cell size s to measure l and initialize $l = 3s$.

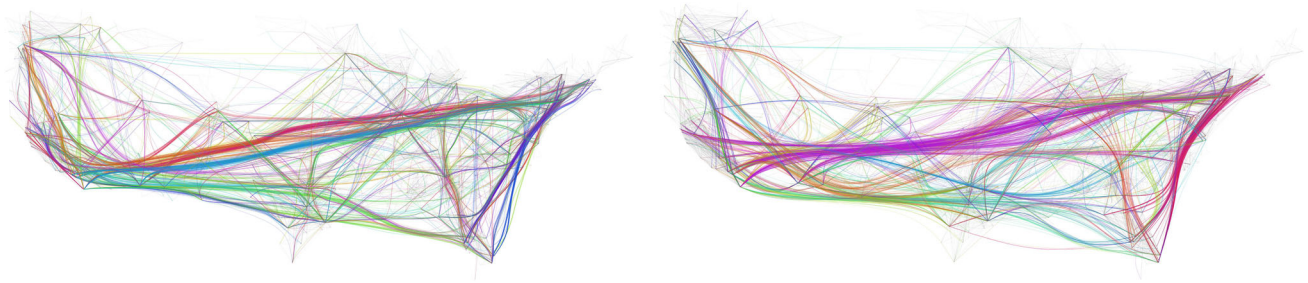
5 Three case studies

We apply LEB to several real-world datasets and demonstrate its effectiveness in achieving the three goals (stated in Sect. 1). The first two datasets, the US migrations and US airlines data, have been used by other edge bundling algorithms and thus serve useful purpose in comparison [1,5,9,14,15,17].

The results in Fig. 9 demonstrates that LEB reveals the backbone well, with or without colors. It also generates distinguishable, smooth and long bundles with few tangled intersections. Comparing with the previous approaches, LEB produces fewer "hubs", or highly concentrated areas, and thus more traceable.

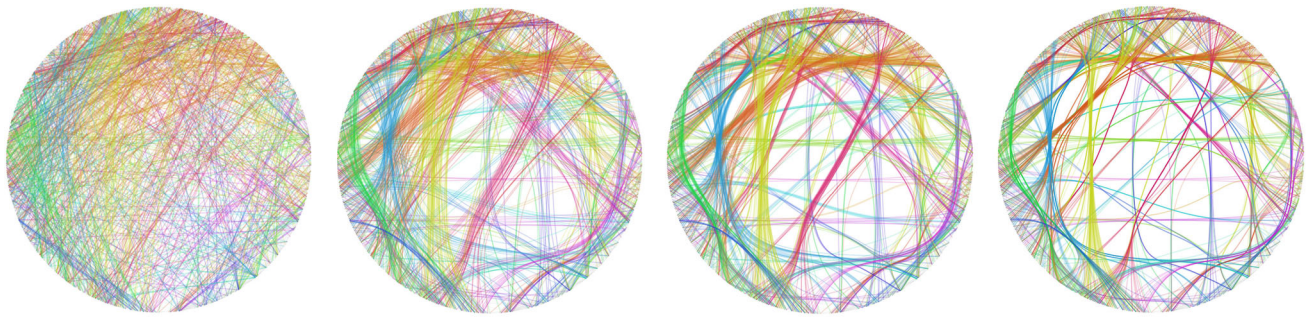
In Fig. 9a and b, there is a major bundle in the east coast and one in the west coast. The east-west long bundles are almost straight, leaving plenty of space for labeling and other application marking. The most striking example is the top bundle (from Portland to Minneapolis), shown in none of the previous approaches.

With the US airlines dataset in Fig. 9c and d, three major east-west flight routes are shown as orange, blue and green bundles. The views could identify major air hubs, even the ones in the middle (west and southwest), such as Salt Lake City and Houston. Different from the previous approaches, LEB generates mutually non-interfering bundles, even those intersecting bundles appear not to influence each other, such as the deep blue bundle (from Detroit and Cleveland to Florida peninsula) and the purple bundle (from Houston to Philadelphia, New York and Boston via Atlanta) in Fig. 9d.



(a) 40-pixel cells **(b)** 120-pixel cells

Fig. 10 Results with different cell sizes in a 1600 × 800 display space (40 pixels in **a** and 120 pixels in **b**)



(a) **(b)** **(c)** **(d)**

Fig. 11 Results for the InfoVis citation dataset: **a** for unbundled and **b**, **c**, **d** for LEB bundling effects under different tension settings

The viewer could therefore trace a bundle from one end to the other. This is all due to our top-down approach giving long edges more priority.

We also demonstrate in Fig. 10 the impact of the cell size. With increasing cell sizes (Fig. 10a, b), a bundle may contain more edges and become smoother and therefore, the overall result does reveal high-level patterns. For example, two long bundles in the middle of Fig. 10a are merged into one and become more traceable with larger cells in Fig. 10b.

To evaluate LEB's effectiveness in visualizing abstract graphs with flexible node positions, we also run LEB with the InfoVis citation dataset, as shown in Fig. 11 where papers are placed on the ring, ordered chronologically and clockwise from 12 o'clock. Figure 11a layouts the graph using straight lines without edge bundling and thus shows serious clutter. Figure 11b–d demonstrate the bundling effects under different tension settings determined by the coefficients of the B-Spline algorithm (see Sect. 4). By automatically adjusting the coefficients and animating different bundling results, the user could select a desirable setting for his/her application. Figure 11d shows an interesting pattern with concentration of citations after every few years (approximate 4 to 6) with several parallel bundles. This could imply the cyclic nature of hot topics and/or typical citation delays.

The first and second rows of Table 1 list the major statistics of these three datasets.

6 Implementation

Using web-based techniques, we have implemented the LEB approach in JavaScript. Our implementation gains a significant speed-up by reusing the routed paths. If two or more edges have an identical path on the grid, we can calculate the path once and reuse it for subsequent routing. We ran LEB on a MacBook Pro with an Intel Core i5 2.6 GHz, 8 GB of RAM, a display of 1650 × 1080 resolution, and an Intel graphics card. All time measurements were averaged on ten runs. All parameters were initialized as described in Sect. 4.

Table 1 gives the performance statistics on the three datasets, including the elapsed times of the six steps with their split percentage. We summarize the performance as the following. (1) Our approach supports real-time interaction, as shown in the attached supplemental video clip. (2) The routing and coloring steps account for the majority of the execution time, 94.3% for the US airlines dataset, 75.3% for the US migrations dataset and 74.8% for the citations dataset. (3) The execution times of Steps 1 and 2 almost linearly increase with $|E^*|$[1] . The average complexity of the coloring step is inversely proportional to the number of identified bundles instead of $|E^*|$ or $|V|$, far lower than its worst-case complexity. Due to fewer identifiable bundles in the US airlines

[1] E^* is E without short edges.

Table 1 Data size and performance statistics

Steps	US airlines	US migrations	InfoVis citation
$\|V\|$	235	1715	512
$\|E\|$	2101	9780	1553
Primary directioning	9.4(0.4%)	34.3(1.9%)	7.4 (1.1%)
Space partitioning	18.6 (0.8%)	43.6 (2.4%)	21.2 (3.2%)
Routing	592.4 (25.5%)	549.8 (30.3%)	345.4 (52.7%)
Smoothing	3.0 (0.1%)	2.8 (0.1%)	2.2 (0.3%)
Coloring	1598.4 (68.8%)	817.1 (45.0%)	145.2 (22.1%)
Rendering	102.0 (4.4%)	367.3 (20.2%)	134.0 (20.4%)
Total (ms.)	2323.8	1814.9	655.4

dataset, coloring takes more time than that for the US migrations dataset. (4) The reuse technique significantly improves the performance, since the number of the edges needed to be rerouted decreases quickly as more edges are rerouted. Although the US migrations dataset is about 5 times larger than the US airlines dataset, it is faster than the latter, thanks to reuse.

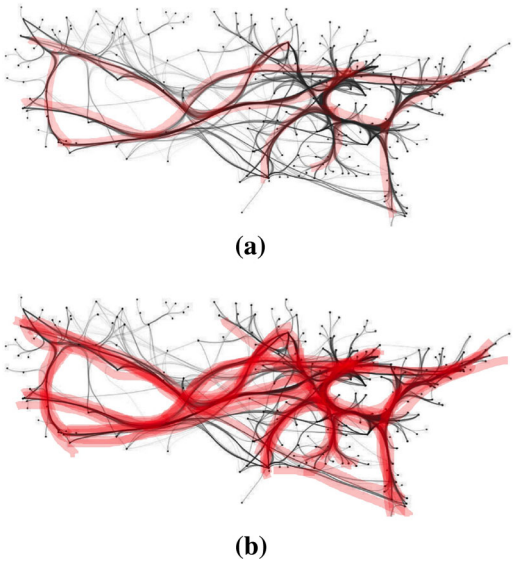
7 Evaluation

To evaluate and compare our LEB with other edge bundling approaches, we have conducted a user study, in which subjects were asked to identify the backbones of the visualizations generated by different approaches. There has been no evaluation for edge bundling effects.

(h) Methods As mentioned in Sect. 1, edge bundling should aim at revealing the backbone of the underlying graphic data and generating traceable bundles. Our user study therefore focuses on the evaluation of various edge bundling approaches in these aspects. We design the task to estimate the overview rather than to obtain accurate answers [18], as described in "Setup and Task" below.

We collected 20 visualization images generated by seven different edge bundling approaches (including ours) on the US migrations and US airlines datasets from the published papers and authors' home pages. These seven approaches are: GBEB [1], FDEB [14], WR [17], MINGLE [9], SBEB [5], KDEEB [15] and our LEB. The images' resolutions are at least 1000×600 so no visible stair-casing effect. Some are colored while others are black and white (B/W). For each colored image, we generated a B/W counterpart and tested both images on the subjects.

(i) Participants A total of 25 graduate and undergraduate students (12 female and 13 male), majoring in computer science, software engineering and visual arts, participated in the study as experimental subjects. Each subject performed 10 trials. One subject's result is considered invalid and thus removed from the report, as it shows strokes mostly in blank



(a)

(b)

Fig. 12 Examples of subjects' drawn strokes and superimposition: **a** strokes by one subject on the B/W image produced by SBEB [5] on the US airlines dataset and **b** corresponding superimposed strokes by 12 subjects

parts of the given images (possibly misunderstood the task). As a result, a total of 240 ($= 24 \times 10$) valid results are reported and each visualization image includes 12 ($= 240/20$) trials.

(j) Apparatus The experiment was conducted on a 10-inch iPad Air 2 displaying a visualization image at a time.

(k) Procedure After entering relevant personal information, subjects read a brief introduction on the task and a 11-second stroke-drawing demonstration in an animated GIF format. Of 20 images generated from LEB and six other approaches, 10 images were randomly selected and displayed in a random order. The subjects were asked to draw the backbone of each displayed image in up to eight strokes (straight or curved) using a finger. The subjects took an average of 6 to 8 minutes to perform the task.

(l) Experimental sesign Figure 12a shows a typical trial result and Fig. 12b shows the superimposed strokes by 12 subjects, with opacity 0.5. Having superimposed the strokes

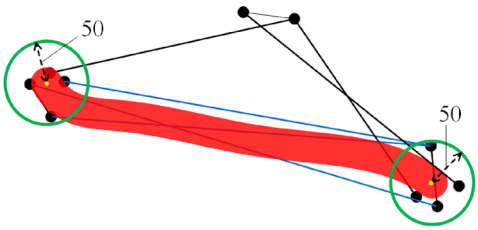


Fig. 13 A subject's drawn stroke (red) and its covered edges (blue) and uncovered edges (black)

by the subjects on the same image, we find that some bundles are identified by almost all the subjects, while others are not. To measure whether a bundle is identified, considering the inaccuracy of finger-touch selection, we draw a circle of 50-pixel radius around each of the two endpoints (yellow dots) of a stroke i (red thick line in Fig. 13). Circles of 20-pixel and 100-pixel radii have also been tested and produce the same ranking order. If an edge's both endpoints fall into the two circles, we consider the edge is identified (e.g., blue lines in Fig. 13) and counted in the total number of identified edges E_i for the ith stroke. We then compute the percentage of identified edges (coverage) over total number of edges E^* for all the strokes on an image j in trial p:

$$C_j^p = \sum_{i=1}^{i=8} E_i / E^* \times 100 \tag{6}$$

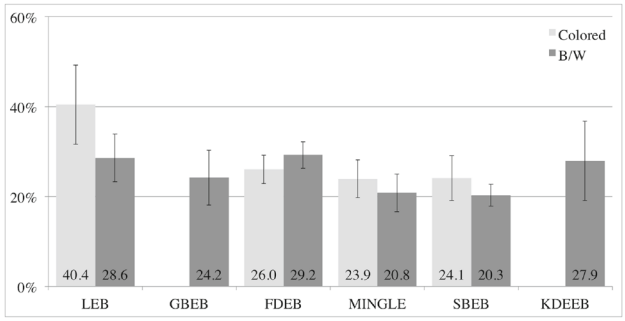
Each image includes 12 trials, the mean coverage and standard deviation for image j are:

$$\text{Mean}_j = \sum_{p=1}^{p=12} C_j^p / 12, \text{ and}$$

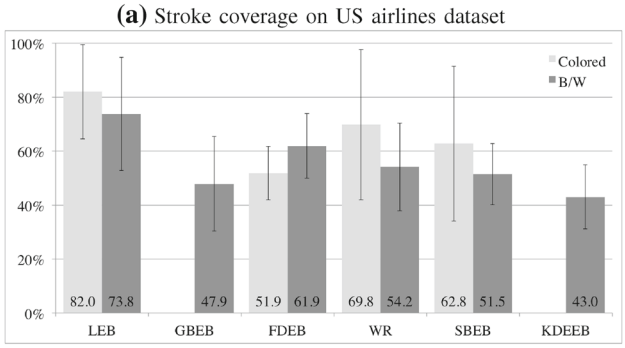
$$\text{SD}_j = \sqrt{\sum_{p=1}^{p=12} (C_j^p - \text{Mean}_j)^2 / 12} \tag{7}$$

Since short edges seriously perturb the evaluation, the number of short edges removed in Sect. 3-f does not contribute to the overall count of edges E. We therefore use E^*, equally E minus the number of short edges in the above equations. Figure 14 shows the 20 means and standard deviations for all the seven approaches, on the US airlines dataset and the US migrations dataset.

(m) Data analysis We summarize the evaluation results in the following. (1) In general, though with a higher variance, color images outperform B/W ones in the evaluation. This implies that color differentiates bundles well. (2) Subjects covered more edges on images for the US migrations dataset than those for the US airlines dataset. We believe that this is partly due to clearer patterns in the US migrations



(a) Stroke coverage on US airlines dataset



(b) Stroke coverage on US migrations dataset

Fig. 14 Coverage percentage of strokes on bundled edges over all edges on **a** US airlines dataset and **b** US migrations dataset (light gray bars for colored images and dark gray bars for B/W images)

dataset. For example, routes in the Great Lakes and Washington areas are hard to be identified by a few strokes on the images for the US airlines dataset. (3) Among all the edge bundling approaches, LEB performs the best for both two datasets and particularly achieves the highest coverage rates for the colored images. (4) The performances for the US airlines dataset show little difference among all the approaches, possibly due to the characteristics of this dataset.

We also perform the one-way ANOVA analysis and Tukey's HSD (Honestly Significantly Different) post-hoc test to determine whether there were significant differences between the 20 visualization images on the two datasets. On both two datasets, the ANOVA analysis indicates that the differences in coverage rates between the ten visualization images are statistically significant ($F(9, 110) = 3.9$, $p = 0.0003 < 0.05$ for US airlines dataset and $F(9, 110) = 2.14$, $p = 0.0318 < 0.05$ for US migrations dataset). The Tukey's HSD post-hoc test on a pairware comparison with $\alpha = 0.1$ shows that on US airlines dataset, (1) "colored LEB" significantly differs from (better than) "B/W GBEB", "colored FDEB", "B/W SBEB", "colored SBEB" and "B/W KDEEB"; (2) other pairs do not show significant differences. On US migrations dataset, (3) "colored LEB" significantly differs from (better than) "B/W SBEB" and "B/W KDEEB"; (4) other pairs do not show significant differences. These results support again that color enhances LEB's performance.

Although "B/W LEB" has first or second highest mean coverage rates on two datasets, the differences between it and others are not significant due to the high variances.

8 Conclusion

This paper has presented a parameterizable 6-step edge bundling approach called LEB that could reveal the backbone structure of any large graph, with distinguishable and traceable edge bundles. Our experiments with the US migrations and US airlines datasets (both having fixed node positions) have shown that LEB outperforms other edge bundling approaches. By reusing the routing computation, LEB delivers a quite satisfactory running speed that is fast enough to support real-time interaction and animation. The LEB approach with six steps has of course room for further optimization and speed-up, which will be our immediate future work. We plan to parallelize the routing step using multithreading and/or GPU techniques.

By experimenting with the InfoVis citation graph, whose nodes can be flexibly placed anywhere, we find LEB to be generally applicable to any node-link or networking diagrams and provide customizable edge bundling visualizations. We plan to further evaluate the effectiveness of LEB in visualizing node-link graphs with millions of nodes and edges.

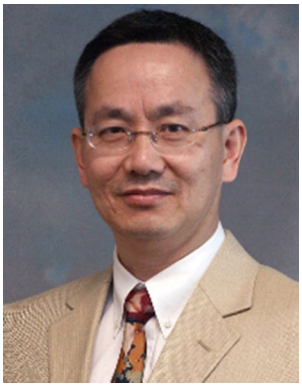
There are several limitations in our approach. As control points are fixed at the centers of cells, undesirable local bends may still exist. We plan to add a control point adjustment step to minimize this. Depending upon the data characteristics, some edges may not be assigned to any primary direction due to their small numbers in their own directions. Our current solution is to assign them to their nearest primary directions. As a future work, we may treat them differently as outliers, and possibly assign them with different colors or investigating a better solution. The high variances in our experiment may imply that the task and measurement method we used should be improved for more stable results, and we will implement this in the future.

References

1. Cui, W., Zhou, H., Qu, H., Wong, P.C., Li, X.: Geometry-based edge clustering for graph visualization. IEEE Trans. Vis. Comput. Graph. **14**(6), 1277–1284 (2008)
2. Dechter, R., Pearl, J.: Generalized best-first search strategies and the optimality of A*. J. ACM (JACM) **32**(3), 505–536 (1985)
3. Doane, D.P.: Aesthetic frequency classifications. Am. Stat. **30**(4), 181–183 (1976)
4. Ellis, G., Dix, A.: A taxonomy of clutter reduction for information visualisation. IEEE Trans. Vis. Comput. Graph. **13**(6), 1216–1223 (2007)
5. Ersoy, O., Hurter, C., Paulovich, F.V., Cantareiro, G., Telea, A.: Skeleton-based edge bundling for graph visualization. IEEE Trans. Vis. Comput. Graph. **17**(12), 2364–2373 (2011)
6. EXCEL 2007: Histogram, (2015)
7. Freedman, D., Diaconis, P.: On the histogram as a density estimator: L2 theory. Probab. Theory Relat. Fields **57**(4), 453–476 (1981)
8. Fua, Y.-H., Ward, M. O., Rundensteiner, E. A.: Hierarchical parallel coordinates for exploration of large datasets. In: Proceedings of Visualization'99, pp. 43–50. IEEE Computer Society Press (1999)
9. Gansner, E. R., Hu, Y., North, S., Scheidegger, C.: Multilevel agglomerative edge bundling for visualizing large graphs. In: IEEE Pacific Visualization Symposium (PacificVis), pp. 187–194. IEEE (2011)
10. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)
11. Hartuv, E., Shamir, R.: A clustering algorithm based on graph connectivity. Inf. Process. Lett. **76**(4), 175–181 (2000)
12. Healey, C. G.: Choosing effective colours for data visualization. In: Proceedings of Visualization'96, pp. 263–270. IEEE (1996)
13. Holten, D.: Hierarchical edge bundles: visualization of adjacency relations in hierarchical data. IEEE Trans. Vis. Comput. Graph. **12**(5), 741–748 (2006)
14. Holten, D., Van Wijk, J. J.: Force-directed edge bundling for graph visualization. In: Computer Graphics Forum, vol. 28, pp. 983–990. Wiley (2009)
15. Hurter, C., Ersoy, O., Telea, A.: Graph bundling by kernel density estimation. In: Computer Graphics Forum, vol. 31, pp. 865–874. Wiley (2012)
16. Koenig, S., Likhachev, M., Liu, Y., Furcy, D.: Incremental heuristic search in AI. AI Mag. **25**(2), 99 (2004)
17. Lambert, A., Bourqui, R., Auber, D.: Winding roads: routing edges into bundles. In: Computer Graphics Forum, vol. 29, pp. 853–862. Wiley (2010)
18. Lee, B., Plaisant, C., Parr, C. S., Fekete, J.-D., Henry, N.: Task taxonomy for graph visualization. In: Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization, pp. 1–5. ACM (2006)
19. Leung, Y.K., Apperley, M.D.: A review and taxonomy of distortion-oriented presentation techniques. ACM Trans. Comput–Human Interact. (TOCHI) **1**(2), 126–160 (1994)
20. Liu, S., Cui, W., Wu, Y., Liu, M.: A survey on information visualization: recent advances and challenges. Vis. Comput. **30**(12), 1373–1393 (2014)
21. Luo, S.-J., Liu, C.-L., Chen, B.-Y., Ma, K.-L.: Ambiguity-free edge-bundling for interactive graph visualization. IEEE Trans. Vis. Comput. Graph. **18**(5), 810–821 (2012)
22. McDonnell, K. T., Mueller, K.: Illustrative parallel coordinates. In: Computer Graphics Forum, vol. 27, pp. 1031–1038. Wiley (2008)
23. Nguyen, Q., Hong, S.-H., Eades, P.: TGI-EB: a new framework for edge bundling integrating topology, geometry and importance. In: Graph Drawing, pp. 123–135. Springer (2012)
24. Peng, D., Lu, N., Chen, W., Peng, Q.: SideKnot: revealing relation patterns for graph visualization. In: IEEE Pacific Visualization Symposium 2012 (PacificVIS), pp. 65–72. IEEE (2012)
25. Peng, W., Ward, M. O., Rundensteiner, E. A.: Clutter reduction in multi-dimensional data visualization using dimension reordering. In: IEEE Symposium on Information Visualization 2004 (INFOVIS 2004), pp. 89–96. IEEE (2004)
26. Phan, D., Xiao, L., Yeh, R., Hanrahan, P.: Flow map layout. In: IEEE Symposium on Information Visualization 2005 (INFOVIS 2005), pp. 219–224. IEEE (2005)
27. Pupyrev, S., Nachmanson, L., Bereg, S., Holroyd, A. E.: Edge routing with ordered bundles. In: Graph Drawing, pp. 136–147. Springer (2012)

28. Qu, H., Zhou, H., Wu, Y.: Controllable and progressive edge clustering for large networks. In: Graph Drawing, pp. 399–404. Springer (2007)
29. Selassie, D., Heller, B., Heer, J.: Divided edge bundling for directional network data. IEEE Trans. Vis. Comput. Graph. **17**(12), 2354–2363 (2011)
30. Shimazaki, H., Shinomoto, S.: A method for selecting the bin size of a time histogram. Neural Comput. **19**(6), 1503–1527 (2007)
31. Sturges, H.A.: The choice of a class interval. J. Am. Stat. Assoc. **21**(153), 65–66 (1926)
32. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972)
33. Telea, A., Ersoy, O.: Image-based edge bundles: simplified visualization of large graphs. In: Computer Graphics Forum, vol. 29, pp. 843–852. Wiley (2010)
34. Tennekes, M., de Jonge, E.: Tree colors: color schemes for tree-structured data. IEEE Trans. Vis. Comput. Graph. **20**(12), 2072–2081 (2014)
35. Verbeek, K., Buchin, K., Speckmann, B.: Flow map layout via spiral trees. IEEE Trans. Vis. Comput. Graph. **17**(12), 2536–2544 (2011)
36. Wong, N., Carpendale, S., Greenberg, S.: Edgelens: an interactive method for managing edge congestion in graphs. In: IEEE Symposium on Information Visualization 2003 (INFOVIS 2003), pp. 51–58. IEEE (2003)
37. Zhou, H., Xu, P., Yuan, X., Qu, H.: Edge bundling in information visualization. Tsinghua Sci. Technol. **18**(2), 145–156 (2013)
38. Zhou, H., Yuan, X., Cui, W., Qu, H., Chen, B.: Energy-based hierarchical edge clustering of graphs. In: IEEE Pacific Visualization Symposium 2008 (PacificVIS), pp. 55–61. IEEE (2008)
39. Zhou, H., Yuan, X., Qu, H., Cui, W., Chen, B.: Visual clustering in parallel coordinates. In: Computer Graphics Forum, vol. 27, pp. 1047–1054. Wiley (2008)
40. Zinsmaier, M., Brandes, U., Deussen, O., Strobelt, H.: Interactive level-of-detail rendering of large graphs. IEEE Trans. Vis. Comput. Graph. **18**(12), 2486–2495 (2012)

Zhuang Cai obtained his M.Sc. in software engineering from Tianjin University, China, in 2016 and his B.Sc. from the same university in 2013. He is now an AI software engineer at Mobike, Inc.



Kang Zhang is Professor and Director of Visual Computing Lab, Department of Computer Science at the University of Texas at Dallas. He received his B.Eng. in Computer Engineering from University of Electronic Science and Technology of China in 1982, Ph.D. from the University of Brighton, UK, in 1990, and Executive MBA from the University of Texas at Dallas in 2011. Prior to joining UT-Dallas, he held academic positions in the UK, Australia, and China. Dr. Zhang's current research interests include visual languages, aesthetic computing, generative art, and software engineering and has published over 250 papers in these areas. He has authored and edited seven books. Dr. Zhang is an ACM Distinguished Speaker, and on the Editorial Boards of Journal of Big Data, The Visual Computer, Journal of Visual Languages and Computing, International Journal of Software Engineering and Knowledge Engineering, and International Journal of Advanced Intelligence. His home page is at www.utdallas.edu/~kzhang.

Dong-Ni Hu obtained her M.Sc. in software engineering from Tianjin University, China, in 2017 and her B.Sc. from Taiyuan University of Technology, China, in 2014. She is now a data analyst at China Mobile Limited.