HAL
open science

Interactive Structure-aware Blending of Diverse Edge Bundling Visualizations

Yunhai Wang, Mingliang Xue, Yanyan Wang, Xinyuan Yan, Baoquan Chen, Chi-Wing Fu, Christophe Hurter

To cite this version:

Yunhai Wang, Mingliang Xue, Yanyan Wang, Xinyuan Yan, Baoquan Chen, et al.. Interactive Structure-aware Blending of Diverse Edge Bundling Visualizations. IEEE Transactions on Visualization and Computer Graphics, 2020, 26 (1), pp.687-696. 10.1109/TVCG.2019.2934805 . hal-02917109

HAL Id: hal-02917109

https://enac.hal.science/hal-02917109v1

Submitted on 5 Jan 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Interactive Structure-aware Blending of Diverse Edge Bundling Visualizations

Yunhai Wang, Mingliang Xue, Yanyan Wang, Xinyuan Yan,
Baoquan Chen, Chi-Wing Fu, Christophe Hurter

Abstract— Many edge bundling techniques (i.e., data simplification as a support for data visualization and decision making) exist but they are not directly applicable to any kind of dataset and their parameters are often too abstract and difficult to set up. As a result, this hinders the user ability to create efficient aggregated visualizations. To address these issues, we investigated a novel way of handling visual aggregation with a task-driven and user-centered approach. Given a graph, our approach produces a decluttered view as follows: first, the user investigates different edge bundling results and specifies areas, where certain edge bundling techniques would provide user-desired results. Second, our system then computes a smooth and structural preserving transition between these specified areas. Lastly, the user can further fine-tune the global visualization with a direct manipulation technique to remove the local ambiguity and to apply different visual deformations. In this paper, we provide details for our design rationale and implementation. Also, we show how our algorithm gives more suitable results compared to current edge bundling techniques, and in the end, we provide concrete instances of usages, where the algorithm combines various edge bundling results to support diverse data exploration and visualizations.

Index Terms—path visualization, trajectory visualization, edge bundles

1 Introduction

Edge bundling techniques have received great interest as a data simplification tool for supporting decision making and data representation [29]. Such techniques help avoid clutter in dense trail-sets and graphs to enhance the visual understanding and exploration.

While many edge bundling techniques exist, it is often challenging for users to produce desirable aggregation results. Some algorithms are too specific to a certain kind of data like small graphs [14] or large datasets [11, 28]. Large graph aggregation methods try to extract hidden global structures, whereas small dataset aggregation methods address readability issues. Other algorithms provide good results with hierarchical data [13], while some are more generally applicable [17]. Besides dealing with this plethora of algorithms, users often struggle to find suitable algorithm parameters to produce effective visual simplifications. Parameter modification often has unpredictable impact on the final edge bundling results and each algorithm produces different visual results with different aggregation parameters. It remains extremely hard to find global parameters that best fit user needs for diverse visualizations. For instance, dense areas require stronger aggregation, while sparse ones need less visual simplification to ensure their readability. Overall, existing techniques are numerous but difficult to understand with complex parameter settings. This hinders the user ability to create desired aggregation results that fulfill the user requirements.

To address these issues, we investigated a novel way to handle visual aggregations with a task-driven [27] and user-centered approach. We do not consider our approach as a new edge bundling technique, but rather as an effective means to take advantage of the existing ones. Since some edge bundling techniques with their associated parameters can produce suitable local visual results, we explored how to combine these local (good) results together into a single visualization. Therefore, we developed an algorithm to produce a coherent (preserving the dataset visual structure), smooth (avoiding significant distortion), and unambiguous (easing readability) transition between these local visual aggregations. Based on these requirements, we summarize our design rationale for our blending algorithm as follows:

- **DR1:** preservation of the inherent visual data structure,
- **DR2:** smooth transition between different aggregated areas to avoid large distortion,
- **DR3:** ensure readability of the graph structures, especially for the regions of interest, and
- **DR4:** flexible fine-tuning of the end result.

Our approach aims to produce decluttered view for graph analysis. It works as follows: the user first explores different edge bundling techniques to generate aggregation results for a given graph, and specifies areas, where specific edge bundling techniques provide good results. Second, our system then computes a smooth, structural preserving, and ambiguity-free transition between these specified areas. Lastly, the user can further apply local edge deformation to fine-tune the end result.

In summary, the main contributions of this paper are as follows.

- We present the first technique to combine edge bundling results for efficient and coherent data exploration and visualizations. The key contribution is a novel blending method formulated based on a global optimization to combine multiple edge bundling results in a smooth, coherent and ambiguity-free fashion.
- We present three extensions to enable users to further explore the blended edge bundling visualizations: (i) edge vector interpolation to adjust the bundling degree in different regions; (ii) customized aggregation to smoothly combine the bundling of local regions or the links between the source and destination nodes with the global structure; and (iii) directly fine-tuning the blended results by interactively manipulating the edges of interest.
- Lastly, we demonstrate the usefulness of our techniques through three use cases, and develop an efficient GPU-based implementation, where users can interactively blend and edit graphs as large as 50K control points in one second.

2 Related Work

In this section, we first review previous work on edge bundling techniques and discuss existing advanced graph visualization techniques.

- *Y. Wang, M. Xue, Y. Wang, and X. Yan, and X. Li are with Shandong University. Email: {cloudseawang, xml95007, yanyanwang93, yanxinyuan1}@gmail.com.*
- *B. Chen is with Peking University. E-mail: baoquan.chen@gmail.com.*
- *C.-W. Fu is with the Chinese University of Hong Kong and Guangdong Prov. Key Lab. of CV and VR Tech., SIAT. E-mail: cwfu@cse.cuhk.edu.hk.*
- *C. Hurter is with ENAC, France. E-mail: christophe.hurter@enac.fr.*
- *Y. Wang and M. Xue are joint first authors, and C.-W. Fu is the corresponding author.*

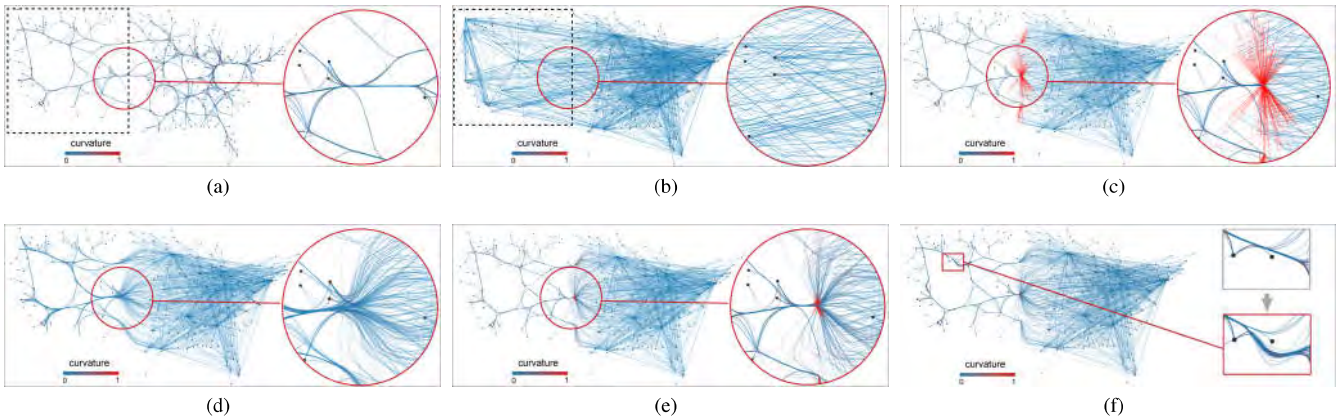

Fig. 1. Transplanting the sub-layout selected by the black box in (a) to the corresponding region of the unbundled layout in (b). (c) Directly replacing the selected regions results in jagged boundaries; (d) smoothing the layout in (c) results in some bundled structures missed; (e) result produced by our method using the constraints of structural preservation and smoothness; and (f) result produced by our method further using the readability constraints that relieve visual ambiguity, where the upper and bottom boxes highlight the result changes before and after applying this constraint.

2.1 Edge Bundling Techniques

Many types of edge bundling techniques exist. A survey proposed a classification based on the type of dataset and on user tasks [29]. In this section, we will highlight the variety of existing algorithms, while stressing their best usage for a given data type and user expectation.

Geometric methods. Some edge bundling techniques are analytically-based, where they first consider the already existing structure of a graph to best display it. This statement best describes the first edge bundling technique, which captures the hierarchical structure of a dataset to display its aggregated visualization [13]. More recent works like the Cactus Tree methods emphasize this type of hierarchical visualization [3] or the common routes [25]. User evaluations have been conducted to assess the best hierarchical simplified visualization for given user tasks [4]. When the statistical inherent data variability of clustered bundled is significant, functional decomposition can be used [18]. Even with a given type of dataset, the abundance of edge bundling techniques might makes it difficult for users to find a suitable one for a given task. This motivates the present study to find more flexible and task-driven edge bundling techniques.

Pixel-based methods. Recent advances in edge bundling helped to visually simplify large datasets [17]. Pixel-based methods do not directly consider the inherent structure of the data but they make it emerge thanks to data density computation [40]. Advanced GPU technique helps overcome the data size limit [48] with a streaming [28] process. As such, these techniques best fit large datasets. While these techniques are fast [21], their parameters are considered too abstract, and it remains difficult and even impossible to find edge bundling parameters that best fit every aggregated area in a single dataset. This motivates the present study rationale towards readability issues in some aggregated areas.

2.2 Advanced Graph Visualization

To produce high quality graph visualizations, many different techniques have been proposed [50]. Here, we briefly review those closest to our work on constrained graph layout, improving graph readability, and smooth visual transition.

Constrained graph layout. Many graph layout techniques exist, such as the traditional spring-electrical model [8, 10] or the stress model [22]. However, most cannot meet some aesthetic metrics [35,36], e.g., minimizing edge crossings and minimizing node overlap. To address this problem and enhance the flexibility, a variety of constrained graph layout techniques have been proposed [6,7,52]. Among them, Wang et al. [52] reformulated the stress model into an edge-vector-based representation, so that various constraints can be modeled. Since edge vectors enable the users to customize the edge orientation and edge length, they have been extended for supporting structure-aware fisheye views [53]. Likewise, we formulate a new optimization model by using edge vectors to preserve the desired structures, while maintaining the smoothness and improved readability in the final results.

Improving graph readability. Graph and trail-set readability is an issue for data exploration and understanding [38]. For instance, confluent drawing uses a power graph decomposition to ensure readability and a non-ambiguity path following in a bundled dataset [1,43]. Untangling the hairballs of multi-centric, small-world online social media networks [33] has also been studied. Local deformation can be interactively achieved via interactive link curvature in network diagrams [37]. Local deformation can also help in obstacle avoidance and for aesthetics purposes [17]. Our work provides additional tools to support graph and trail-set readability with interactive deformation techniques.

Smooth visual transition. Previous works studied transitions between graphs or trail-set layouts. Most existing techniques considered different layouts, where nodes do not have fixed locations [19, 53]. In our work, nodes in the given layouts have fixed locations, and only edges are distorted to ensure a visual simplification. Offline dynamic graph drawing is also a generalization of the smooth transition problem, where the temporal coherence constraint [2,53] is often enforced. Single [44] or multiple lenses [23] can be used for this purpose of using animation and interaction [39,45]. The MoleView [19], which uses a dual layout and animation to show bundled and unbundled transitions within a lens, adds flexibility in edge bundling techniques. However, it does not take into account smooth transitions between the dual layouts and a structure-aware constraint. Kruiger et al. also considered a multi-layer blending of edge bundling techniques [24] but this technique does not support the blending of a local area. To the best of our knowledge, no previous technique has investigated local smooth blending between bundled results, while this paper fills the gap.

3 Seamless Blending of Edge Bundling Results

Given an unbundled graph drawing or trail set $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$ with vertex set $\mathbf{V}$ and edge set $\mathbf{E}$, each edge $e_l \in \mathbf{E}$ connects two nodes in the form of a straight line by default. Edge bundling techniques often pre-sample each edge into a set of control points at equal intervals, where each control point in edge e_l has a position $\mathbf{x}_{l,i} \in \mathbb{R}^2$. In this way, we form a set of edge segments, each in-between a pair of adjacent control points. From now on, we call the control points between vertices on the edges as *edge points* and vertices in $\mathbf{V}$ as *nodes*. Using these notations, we first briefly describe the straightforward solutions for blending edge bundling results then formulate the problem as an optimization task based on the design rationale described in the introduction.

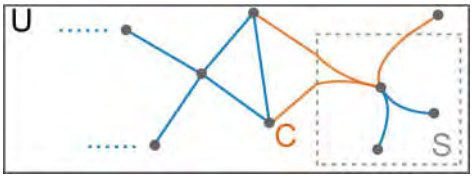

Fig. 2. Illustration of the edge point sets: $\mathbf{U}$ for whole graph; $\mathbf{S}$ for user-selected sub-region; and $\mathbf{C}$ for connecting edges from $\mathbf{U} \setminus \mathbf{S}$ to $\mathbf{S}$.

3.1 Straightforward Solutions

Suppose we have two different bundled results (source and destination layouts) of the same input graph, i.e., same $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$ but different edge point locations. Now, the user wants to specify a sub-region in the source layout and transplant the bundled result inside the sub-region to the destination layout. To facilitate the discussion, we denote $\mathbf{U} = \{(l,i)\}$ as the set of edge point indices for the whole graph, and $\mathbf{S} \subset \mathbf{U}$ as the set of edge point indices in the user-selected sub-region.

To achieve this goal, the simplest and most straightforward way is to replace all $\mathbf{x}_{l,i}$ in destination layout with $\mathbf{x}'_{l,i}$ in source layout for all $(l,i) \in \mathbf{S}$. However, such an approach might not be able to produce smooth layouts, especially for the edges near the boundary between the target sub-region and the remaining region in the layout. Fig. 1 shows an example, where (a) and (b) show the source and destination layouts, respectively. If we directly copy a sub-region from source to destination, we will produce (c), where almost all boundary edges become polylines with large curvatures. Further applying Laplacian smoothing [49] to this result can reduce the sharp boundary edges, but it might significantly change the graph structure, since the bundled structures near the boundary might become unbundled (d). Our method overcomes this issue (e) and further improves readability (f).

Limitations. Though carefully tuning the smoothing parameters might alleviate the structure degradation, the above two-step strategy has three inherent limitations. First, the structures in source and destination layouts often have different degrees of bundling, leading to the challenge of finding adaptive smoothing parameters. For example, the structure of the unbundled part shown in Fig. 1(d) is almost retained, whereas the one in the bundled part is heavily degraded. Second, when two input visualizations have different scales, it is almost impossible to apply a direct replacement because of the different coordinate systems; see an example shown in Fig. 3(a). Lastly, alleviating ambiguities resulting from bundling is a fundamental problem [30], whereas this straightforward solution cannot handle the issue. These limitations motivate the development of a unified framework to produce structure-aware blending of edge bundling results for efficient graph exploration.

3.2 Structure-aware Optimization Framework

To smoothly transplant selected points in $\mathbf{S}$ from source to destination layout, while improving readability, we formulate an optimization framework to solve for edge point positions in the destination layout, $\mathbf{Z} = \{\mathbf{z}_{l,i}\} \forall (l,i) \in \mathbf{U}$, using the following three-fold objective:

$$\begin{aligned} \arg\min_{\mathbf{Z}} & \sum_{(l,i)\in\mathbf{U}\setminus\mathbf{S}} \alpha \, ||\mathbf{z}_{l,i} - \mathbf{x}_{l,i}||^2 \\ & + \sum_{(l,i)\in\mathbf{S}} \beta \, ||\mathbf{z}_{l,i} - \mathbf{z}_{l,i+1} - \mathbf{d}_{l,i}^{l,i+1}||^2 \\ & + \sum_{(l,i)\in\mathbf{C}} \theta \, ||(1-\mu_{l,i})\mathbf{z}_{l,i-1} + \mu_{l,i}\mathbf{z}_{l,i+1} - \mathbf{z}_{l,i}||^2, \end{aligned} \tag{1}$$

where α, β, and θ are weights for each term; $\mathbf{U} \setminus \mathbf{S}$ is the index set of the edge points outside the user-selected sub-region ($\mathbf{S}$); $\mathbf{C}$ is the index set of the edge points on the connecting edges that go into the user-selected sub-region (see Fig. 2 for an illustration; note that we have to optimize the positions of these points for edge smoothness); $\mathbf{x}_{l,i}$ is an edge point position on the l-th edge in the given destination layout; whereas $\mathbf{d}_{l,i}^{l,i+1}$ and $\mu_{l,i}$ are to be defined in the constraints presented in the next subsection. Note also that for edge points outside both $\mathbf{S}$ and $\mathbf{C}$, we simply keep their positions as in the given destination layout according to the first term.

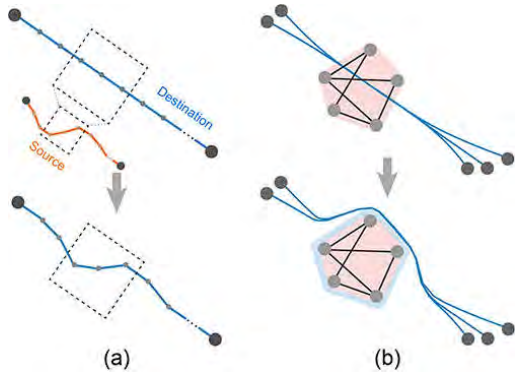

Fig. 3. (a) Blending the edge segments selected by the box (dashed lines) from source to corresponding region in the destination layout. (b) avoiding the occlusion between the blue edge and the pink convex hull by re-routing the blue edges using the readability constraint.

Our optimization model consists of three terms (see Eq. (1)), corresponding to two design rationales (DR1: structure preservation and DR2: smooth transition): the first term penalizes the position change of points outside $\mathbf{S}$; the second term is a vector constraint that maintains the structure of the transplanted structures by preserving the edge orientations of the corresponding edge segments; and the last term enforces the smoothness transition, where the bundles are desired to smoothly transit from the changed region to the unchanged regions.

Regarding the readability constraint in DR3, it is taken as a post-processing step that alleviates the ambiguity of the regions of interest in the final blended result, and thus, Eq. (1) does not have the related term. In the following, we will describe each term in Eq. (1) and how the readability constraint is formulated.

3.3 Structural and Smoothness Constraint

This subsection presents the three terms in Eq. (1) successively:

Position-based constraint. To preserve the unselected points in $\mathbf{U}$, specifically those connected to the edge points in $\mathbf{S}$ on the same edge, we penalize the position change by using the first term in Eq. (1).

Edge-vector-based constraint. We also need to preserve the transplanted structures, where the position-based constraint is not in use for two reasons. First, the absolute positions of the edge points are not important for this task, which calls for an intrinsic structure representation. Second, directly inserting the selected regions to the destination layout is meaningless, if the two layouts have different scales. Here, we introduce the edge-vector-based constraint by defining

$$\mathbf{d}_{l,i}^{l,i+1} = d \frac{\mathbf{x}'_{l,i} - \mathbf{x}'_{l,i+1}}{||\mathbf{x}'_{l,i} - \mathbf{x}'_{l,i+1}||}, \tag{2}$$

where $\mathbf{x}'_{l,i}$ is the position of an edge point in the source layout and d is the edge segment length in the destination layout. In this way, the transplanted edges can naturally adapt to the scale of the destination layout, while preserving the original direction. Fig. 3(a) shows an example, where the source and destination edges have different scales, and yet, our approach can consistently combine them and produce the result shown at the bottom of Fig. 3(a).

Smoothness constraint. To follow DR3, we define the smoothness constraint by using the Laplacian operator:

$$\mathbf{z}_{l,i}^{t} = \frac{(\mathbf{z}_{l,i-1}^{t-1} + \mathbf{z}_{l,i}^{t-1} + \mathbf{z}_{l,i+1}^{t-1})}{3},$$

where $\mathbf{z}_{l,i}^{t}$ denotes the position of the i-th node in the l-th edge at the t-th iteration. However, this scheme assumes all edge segments have

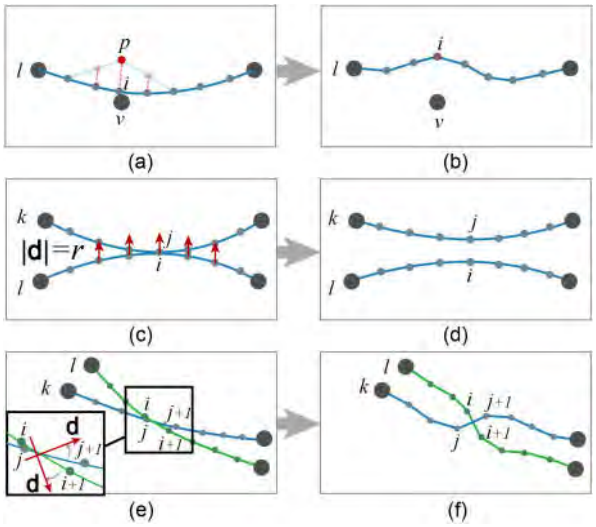

Fig. 4. Improving graph readability by minimizing (a,b) node-edge occlusion and (c,d) edge ambiguity, while maximizing (e,f) the small crossing angle. (a,b) The occlusion between v and the edge l can be solved by displacing point i and its neighbors towards target position $\mathbf{p}$; (c,d) the ambiguity between edges k and l can be resolved by displacing few nearest points with vectors $\mathbf{d}$; and (e,f) maximizing the edge crossing angle by rotating the related edge segments.

the same density, which is unlikely the case for layouts with different degrees of bundling. Accordingly, we define a weight $\mu_{l,i}$ to take into account the length difference between adjacent edge segments:

$$\mu_{l,i} = ||\mathbf{x}_{l,i} - \mathbf{x}_{l,i-1}||/||\mathbf{x}_{l,i+1} - \mathbf{x}_{l,i-1}|| \,, \tag{3}$$

and formulate the smoothness term in the third term of Eq. (1).

Fig. 1(e) shows a result produced by using the three terms in Eq. (1). We can see that the transplanted structure taken from Fig. 1(a) can seamlessly blend with the unbundled layout in Fig. 1(b), where only a small amount of edge points have large curvatures, since they are at the intersections between many differently-oriented edges.

3.4 Readability Constraint

Graph readability is affected by three major problems [51]: (i) node-edge occlusion, (ii) edge ambiguity, and (iii) small edge crossing angles (see Fig. 4 (a,c,e)). As highlighted by the top black box in Fig. 1(f), the bundled structure in our blended result also suffers from these problems. To relieve the first problem, Luo et al. [30] detect node-edge occlusion and then route the edge away from the unrelated nodes during the edge bundling. Since detecting ambiguity in the whole graph is very expensive, this technique is mainly applied to small subgraphs of interest shown in another detailed view. In this work, we extend our optimization framework to improve the readability of regions of interest so as to respect the design rationale DR3 (ensure readability of the graph structures).

In analogy to the blending task, $\mathbf{S}$ is the edge point set in the region of interest and $\mathbf{U}$ is the entire edge point set. As illustrated in Fig. 4, to address these readability problems, we have to manipulate the related edge points of the edges inside or around the user-selected region. To do so, we can readily customize $\mathbf{x}_{l,i}$ and $\mathbf{d}_{l,i}^{k,j}$ in our optimization framework (Eq. (1)) for each case as shown in Fig. 4.

Node-Edge occlusion. For each edge e_l in the region of interest, we follow Luo et al. [30] to detect if e_l passes near any unrelated node in the graph; if it does, we compute the target position ($\mathbf{p}$) for the related edge point on e_l (see Fig. 4(a)), map the neighbors of the edge point to the positions around $\mathbf{p}$, and set them as $\mathbf{x}_{l,i}$ using the first term in Eq. (1). Fig. 4(a) shows an example, where node v is too close to edge l. After applying our optimization, the occlusion can be avoided in the refined result (see Fig. 4(b)), while the edge is smooth.

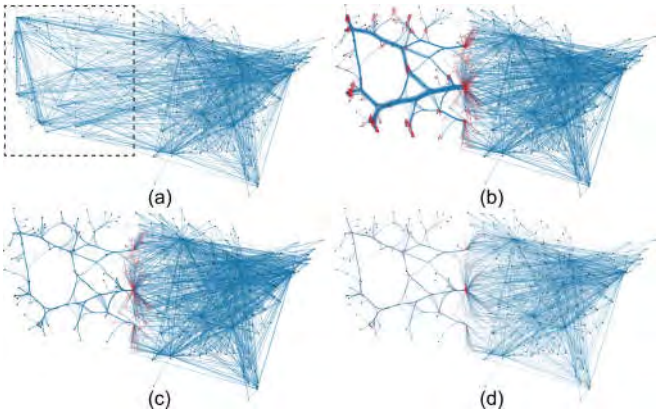

Fig. 5. Convergence of our method. (a) Input layout, and (b)-(d) results optimized after 5, 10, and 30 iterations, respectively.

Node v can also be generalized as a complex cluster, represented by a convex hull, where readability might reduce when edges go through the hull of the cluster. Fig. 3(b) shows such an example, where the unrelated blue edges might distract the user's attention. To address this issue, we project these edges to the boundary of the convex hull and use the projected edge point as $\mathbf{x}_{l,i}$ in the first term of Eq. (1). The bottom part of Fig. 3(b) shows the optimized result, where the blue edges can be re-routed to avoid the convex hull.

Edge congestion. Fig. 4(c) shows an edge congestion case, where edge connectivity becomes ambiguous. To this end, we find edge $e_k (k \neq l)$ for each edge point $\mathbf{x}_{l,i}$ that meets the following two requirements: (i) edges e_k and e_l neither cross each other nor have the same endpoints, otherwise they should be bundled together; and (ii) there exists a point (say the j-th point) on edge e_k, such that the distance between $\mathbf{x}_{l,i}$ and $\mathbf{x}_{k,j}$ is smaller than a given threshold r. If such point $\mathbf{x}_{k,j}$ is found, we set its corresponding $\mathbf{d}_{l,i}^{k,j}$ in Eq. (1) by

$$\mathbf{d}_{l,i}^{k,j} = r\frac{\mathbf{x}_{k,j} - \mathbf{x}_{l,i}}{||\mathbf{x}_{k,j} - \mathbf{x}_{l,i}||}. \tag{4}$$

As shown in Figs. 4(c, d), assigning such vectors to the relevant points on e_k (and also e_l) in the second term of Eq. (1) can displace the two edges away from each other to resolve the ambiguity.

Edge crossing angle. When two edges cross each other, having a large crossing angle leads to better graph readability [15]. Hence, we maximize the crossing angle for such edge pairs in the region of interest. To do so, we first find the intersecting edge pairs then calculate the edge crossing angles ω between them. If ω is smaller than a threshold ϕ ($\phi = \pi/6$ by default), we compute the orientation of the related edge segments, like the two edge segments defined by points $\{\, \mathbf{x}_{l,i}, \mathbf{x}_{l,i+1} \,\}$ on e_l and $\{\, \mathbf{x}_{k,j}, \mathbf{x}_{k,j+1} \,\}$ on e_k, we set their corresponding vectors in Eq. (1) to improve their readability (see also Fig. 4(e)) as

$$\mathbf{d}_{l,i}^{l,i+1} = (\mathbf{x}_{l,i+1} - \mathbf{x}_{l,i}) \oplus (\pi/4 - \omega/2) \tag{5}$$

$$\mathbf{d}_{k,j}^{k,j+1} = (\mathbf{x}_{k,j+1} - \mathbf{x}_{k,j}) \ominus (\pi/4 - \omega/2) \,, \tag{6}$$

where $\oplus$ denotes a clockwise and $\ominus$ a counter-clockwise rotation. Figs. 4(e,f) show examples, where the two edges become almost perpendicular to each other, after the optimization.

It can be noted that two angles in Eq. (5) can be defined in different ways for different applications. For example, to explore edges of interest, one would often like to maintain their orientations, while rotating the intersected edges to maximize the crossing angles.

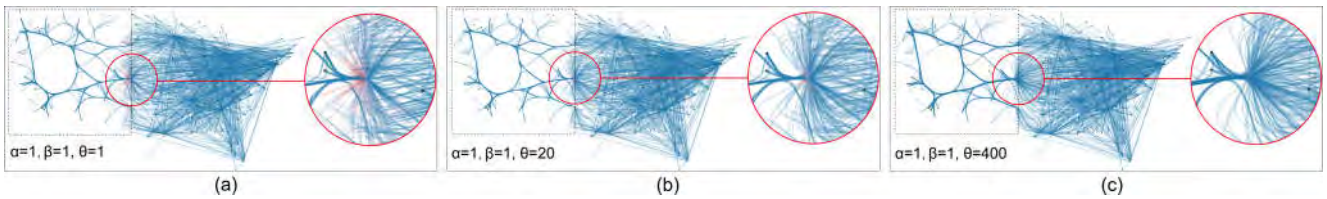

Fig. 6. These three results (a-c) are produced using the same α and β but different θ: 1, 20, and 400, respectively.

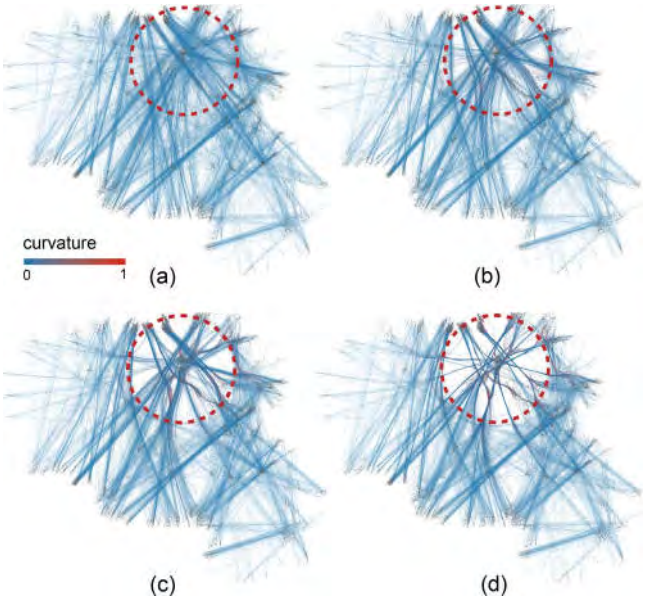

Fig. 7. Edge vector interpolation results in the user-selected region produced using different ω: (a) $\omega = 0$; (b) $\omega = 0.35$; (c) $\omega = 0.75$; and (d) $\omega = 1$.

3.5 Solving the Optimization

To solve for $\mathbf{Z}$ in Eq. (1), we can differentiate Eq. (1) with respect to $\mathbf{Z}$, set the derivative to zero, and produce the following linear system:

$$(\mathbf{M}_P + \mathbf{M}_V + \mathbf{M}_S)\mathbf{Z} = \mathbf{JD} + \mathbf{Q}, \tag{7}$$

where $\mathbf{M}_P$, $\mathbf{M}_V$, $\mathbf{M}_S$ are $n \times n$ weighted Laplacian matrix, representing the parameters for the position-based constraints, vector-based constraints, and smoothness constraints, respectively; n is the number of edge points in $\mathbf{Z}$; matrix $\mathbf{Q}$ represents all the weighted positions of the unselected points in the destination layout and is defined by the first term; both matrices $\mathbf{D}$ and $\mathbf{J}$ are derived from the second term in Eq. (1), where $\mathbf{D}$ stores all the target edge vectors $\mathbf{d}_{l,i}^{l,i+1}$.

Given the user-selected blending areas, we first solve for $\mathbf{Z}$, which represents the edge positions in the blended layout. Then, we incorporate the readability constraints by looping over all the edge points and updating the entries in $\mathbf{x}_{l,i}$ and $\mathbf{d}_{l,i}^{k,j}$, and then solve for $\mathbf{Z}$ again. By using a conjugate gradient solver, the desired solution can be quickly obtained. Fig. 5 shows the convergence process, where the result after 30 iterations already satisfies most of the constraints.

The final results are heavily influenced by the weights of the three terms α, β, and θ in Eq. (1). Since the first and second terms both correspond to DR1 and the third term corresponds to DR3, α and β are assumed to be the same and the ratio between α and θ influences the final result. When the ratio is too small, the result is similar to the one produced by the straightforward solution (see Fig. 6(a)); when the ratio is too large, DR1 might not be fully respected (see Fig. 6(c)). In our experiment, α, β, and θ are empirically set as 1, 1, and 20, respectively; see Fig. 6(b). Note also that both α and β correspond to DR1, so we set α=β, since their ratio influences the final result.

4 Exploration of the Blended Result

In this section, we present three extensions that assist users to interactively blend and edit edge bundling visualizations, for supporting efficient exploration of large graphs and trails.

4.1 Edge Vector Interpolation

A typical interaction for seamless blending of edge bundling results is to combine parts from different results together in a single layout; see an example in Fig. 1. To help users explore the blended results, we introduce *edge vector interpolation* that allows users to generate results with varying degrees of bundling in different regions.

Supposing we have a blended layout generated by fusing two bundled layouts, the vector of a certain edge segment inside the user-selected region is $\mathbf{sd}_{l,i}^{l,i+1}$ in one layout and $\mathbf{td}_{l,i}^{l,i+1}$ in the other. Given the weight ω, the new edge vector of this segment becomes

$$\mathbf{d}_{l,i}^{l,i+1} = \tau * \mathbf{sd}_{l,i}^{l,i+1} + (1-\tau) * \mathbf{td}_{l,i}^{l,i+1}. \tag{8}$$

By taking $\mathbf{d}_{l,i}^{l,i+1}$ as the transplanted edge structures into the second term of Eq. (1) and using the blended results as $\mathbf{x}_{l,i}$ to define the first and third terms, we can produce different degrees of bundling in the user-selected region. Fig. 7 shows a complex example, where the area inside the dotted circle is bundled using different weights, increased from (a) to (d). We can see that the bundles in the dotted circle are getting tighter while smoothly connecting with the unselected parts.

4.2 Customized Aggregation

By combining the method with an existing edge bundling technique, the user can analyze the local structures by bundling the edges within the local regions or between the source and destination nodes.

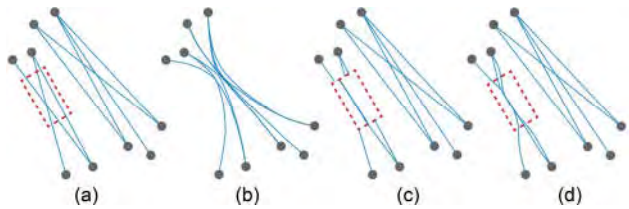

Fig. 8. (a) The unbundled input; (b) the result generated by global bundling, where the two edge sets are mixed together; (c) the result generated by only bundling the segments inside the selected region (note the discontinuity); and (d) our optimized local bundling result.

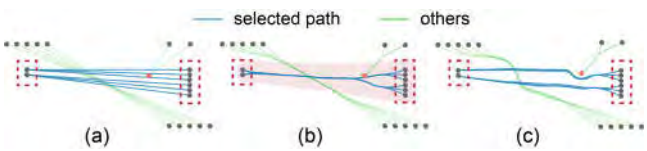

Fig. 9. Path bundling. (a) Input unbundled layout (red boxes are the user-marked origins and destinations); (b) our path bundling result; and (c) our result further enhanced by applying the readability constraints to alleviate the node-edge occlusion, edge congestion and small edge crossing angle issues.

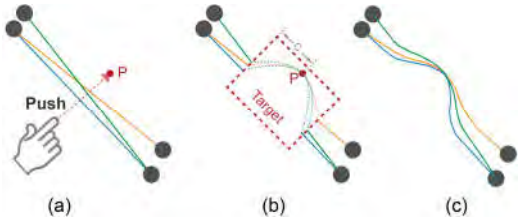

Fig. 10. (a) Pushing the selected edge to position **p**; (b) three Bézier curves generated within distance c from **p**; and (c) the result generated by using our optimization framework.

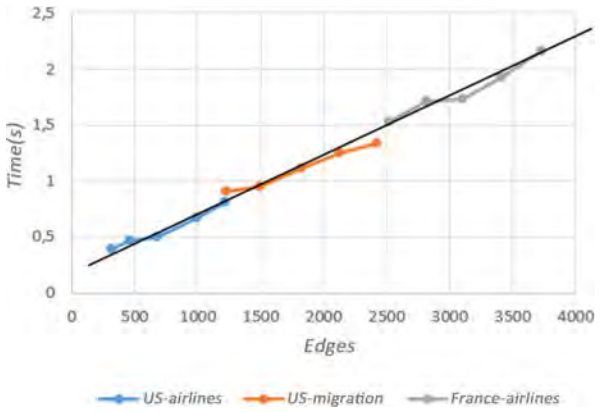

Fig. 11. The above plot reports the time taken by our method to compute the blending between the considered dataset (US-airlines, US-migration, and France-airlines) over different number of edges. Our method remains fast (just a few seconds) even for large datasets (3.5k edges).

Local bundling. Directly bundling the whole graph might miss some local structures, whereas directly bundling the edges of local regions might lead to broken edges. Fig. 8 shows an example, where neither global nor local bundling can preserve the structure of the selected edges. By using the local bundling result to define $\mathbf{d}_{l,i}^{l,i+1}$ in Eq. (1), our optimization can consistently fuse together the local bundling result with the unbundled regions in the layout (see Fig. 8(d)).

Path bundling. Following the paths from specific origins to specific destinations is a common task in exploring trails. To reduce the visual clutter, bundling the related edges in the paths can help users identify the major trends. However, only bundling such edges might be insufficient, especially if there are many intersecting edges along the paths. Accordingly, we extend our method for *path bundling*, which is similar to the graph interaction technique of *from Detail to Overview via Selections and Aggregations* [47].

Specifically, we perform the following procedure to generate path bundling result: (i) given the user-provided origin and destination regions in layout, find all the associated paths as the foreground (see Fig. 9(a)); (ii) bundle all the edge segments along these paths (see Fig. 9(b)); and (iii) further apply the readability constraint (which is optional) to resolve readability issues and reveal the relevant paths that go between the origin and destination areas (see Fig. 9(c)).

4.3 Interactive Manipulation

To help the user interactively fine-tune the bundling results, we introduce the *push* operation, where the target positions for some edge points are specified manually. This interaction adds flexibility for the user to produce visualization results in a direct manipulation style. As shown in Fig. 10, the user can intuitively push the edges towards a target **p**. Here, we define the area within distance c from **p** along the related edges as the target area, and construct a Bézier curve for each associated edge. By using the Bézier curves to define the associated $\mathbf{d}_{l,i}^{l,i+1}$ in our optimization framework, we can readily produce a smooth result; see Fig. 10(c) for an example.

5 Rendering

To add flexibility to our blended results, we adopt an additional post processing step for fine-tuning (DR4). Inspired by the first edge bundling method, which concentrated edges [31] and its extension with Sankey diagrams [46], we apply the following post-processing techniques to improve the rendering style of the aggregated edges.

Density computation and rendering. Sankey diagrams [46] visually encode a specific data dimension using the edge width (i.e., the flow map of French wine exports (Minard, 1864)). Such visual coding has been applied to edge bundling techniques with trails, gazes, and graph data sets, where the edge width depicts the aggregated edge density [34]. Image-based visualization technique [16] can also be applied to improve the displayed edge width. Telea et al. [42] used the distance transformation [41] to emphasize the border of the bundled edges by darkening edges according to their distances to the center line of the cluster being considered. While this technique shows relevant visual improvement, it needs to be applied to each bundle separately, which is computationally challenging for numerous given clusters. To address this issue, we extend these previous techniques to make them flexible for application to complex and dense datasets (see Fig. 12).

First, we compute a density map using kernel density estimation [40]. We use a raster map of 400 by 400 pixels and an 11-pixel-radius Gaussian kernel (other parameters can apply, we provide here a good compromise between computation time and rendering quality). Each edge is then re-sampled with 200 points. Second, we compute the local edge density and average the raster map density value along the considered edge. We empirically use three points before and after the considered local density then map the corresponding average value with the edge width [17]. Finally, we emphasize the edges by drawing a one-pixel-width line on its border.

Trail set ordering and transparency. Visualizing complex and dense datasets might result in heavy visual clutter due to the tangled and overlapping trails/edges. Our computed density field enables us to create varying line widths for each edge segment, where the width of each segment is proportional to its density value. However, simply rendering such width-varying line segments might cause the thick edge segments to be occluded by the thin ones. For this reason, we apply a final edge processing algorithm, where we first sort the edge segments with respect to their local average density and provide user with two rendering options: (i) *high density on top* and (ii) *low density on top*. Fig. 12 shows different types of sorting, where one can better see the main path with high density on top or emphasize small bundles with low density on top. In addition, we allow users to adjust the transparency of the whole graph. With low transparency, more small bundles are shown (see Figs. 12(b,c)); while the main flow is shown with high transparency (see Fig. 12(d)).

6 Implementation

To support an interactive exploration, we implemented a CUDA-based conjugate gradient (CG) solver for solving Eq. (7) and used OpenGL for rendering. For a graph of n edge points, the time complexity of our method depends on the one of the CG algorithm [55], which is $O(n^{1.5})$, while the space complexity is $O(n^2)$. For more implementation detail about CUDA-based CG solver, please refer to Wang et al. [52, 53]. Three edge bundling techniques are employed for graph aggregation, such as kernel density estimation-based edge bundling (KDEEB) [17], FFT based edge bundling (FFTEB) [28], and skeleton-based edge bundling (SBEB) [9]. By running on the NVidia GTX1080 graphics card with 8GB video memory, our system can handle graphs of up to 3.5K edges and 50K edge points in less than three seconds.

Since the computation time of producing initial edge bundling results varies too much [29], we only take into account the bundle blending time to assess the performance of our method. In Fig. 11, we report the

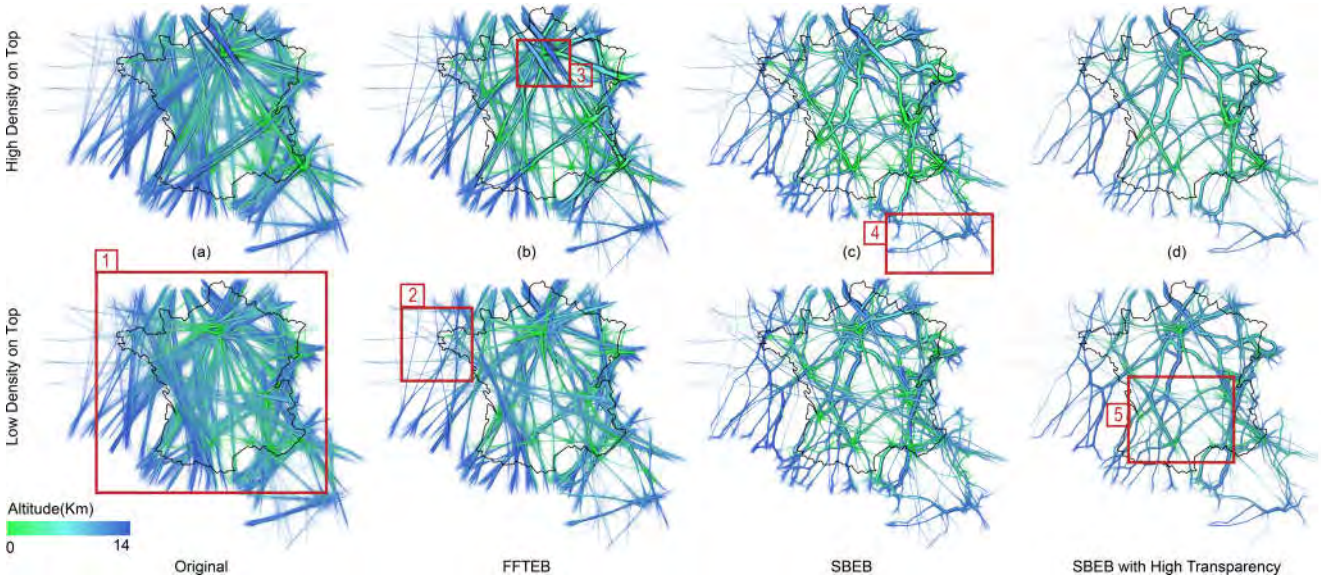

Fig. 12. These visualizations are created using different edge bundling methods and rendering styles for presenting recorded aircraft trajectories over France during one day. From left to right, the visual simplification varies from no simplification to a significant one. The width of edge segments encodes the edge density, and the top row displays the edge segments with high densities on top so that the main bundles are more clearly shown, while the bottom row displays the edges in a reverse way. The last column shows a strong aggregation with high transparency, where the main air flows are more clearly shown. From these various rendering and edge bundling methods, the user can decide which part of each visualization best shows the relevant information and combine them together by using our blending method. For example, the regions with labels "2–5" are selected to blend with the original layout with the label "1" together, formed the result shown in Fig. 13(b).

timing measurements, where three different datasets (US-airlines, US-migration, and France-airlines) with varying numbers of edges were tested. We can see that our method proves to be fast with a computation time of just a few seconds, even for the most complex blending case. Also, Fig. 11 shows that the algorithm run time scales linearly with the number of edges, but we cannot consider that our method can handle any dataset size due to the GPU memory limitation.

7 Applications

In this section we demonstrate the usefulness of the proposed technique to support data exploration and presentation through different use cases. Every example targets the same issue: how to remove local clutter to better understand and read the presented information. The first use case shows how the proposed algorithm can support origin-destination features to perform a local simplification with different edge bundling techniques. The next use case presents the visualization of a trail set and its simplified version with dense and coarse areas. The last example supports the flexibility of our technique with its applicability to different graph deformations: edge bundling and fisheye technique.

7.1 Aircraft Trajectories Composition

In this example, we considered the visualization of one day (Friday, 22nd of February, 2008) of recorded aircraft trajectories over France. This dataset contains 17,851 flight trajectories with 427,651 records. This trail set has been intensively explored in the past with many discoveries in terms of traffic structure and density [16]. Many edge bundling techniques have also been applied but no previous work has investigated how to best visualize various aggregated areas. Considering Fig. 12, some areas with different bundling parameters better display flow specificity: Paris area (box number 3) with average bundling aggregation and high density on the top, west part of France (box number 2) with low density on the top, Corsica area (box number 4) with strong bundling, and center of France (box number 5) with high bundling parameters and high transparency. By using our blending method, we managed to produce a single aggregated view with smooth and ambiguity-free transition between different bundling results and their renderings. Fig. 13 shows the end result, where flow aggregations are more visible compared to the original flow visualization. Our method blends all the parts and insures a smooth transition between them.

To assess the quality in terms of the encoded information, we asked practitioners to comment on Fig. 13. We asked two air traffic controllers (10 and 15 years of experience) to give their opinions regarding the veracity of the aggregated flows and also to compare our composed image with the original one. The main comments were that the aggregated view shows the major air flow but some trails are too distorted compared to the actual ones. The aggregated view is clear with more empty space which eases its comprehension. The Corsica area clearly shows the main flow compared to the original data. The central part of France also better shows the main flows. Overall, the negative comments are generally applicable to the edge bundling issues but there are no complaints about the transition between the various aggregated areas and the resulting view is considered as informative and appealing in terms of visual quality.

7.2 Origin Destination

Origin-Destination (OD) flows show connections between areas of interest on a map. Visualization of such subsets may require visual simplification when the dataset is too large, and thus, create severe overlap [12, 47]. In this example, we show how our algorithm supports the blending of graph aggregations and OD flow filtering.

Fig. 14 shows the investigation of the US-migration dataset. Here, users are allowed to select nodes of interest as origin or destination by adding purple dashed boxes to the original graph (Fig. 14). The user first investigated the original graph (Fig. 14(a)) with the OD flows between two major cities in yellow and a global bundled result (Fig. 14(b)). We can see that the OD flow between the two major cities might be mis-bundled together with other links, especially the ones in the black box, since many overlapped edges in Fig. 14(a) disappear.

To further check if edge congestion results in ambiguity, we applied the path bundling here. Specifically, the algorithm first bundles the OD flow separately, then bundles the edges going through the region of the OD flow shown in the blue box, and finally blend these two bundled results with the fully bundled graph version. This simplified vicinity ensures a better readability of the investigated OD flow. As a final stage, the user can further adjust the OD vicinity transparency to even reduce any unwanted small bundle, and thus, remove extra clutter. Fig. 14(c) shows the final result of our investigated graph with the highlight of a specific OD flow.

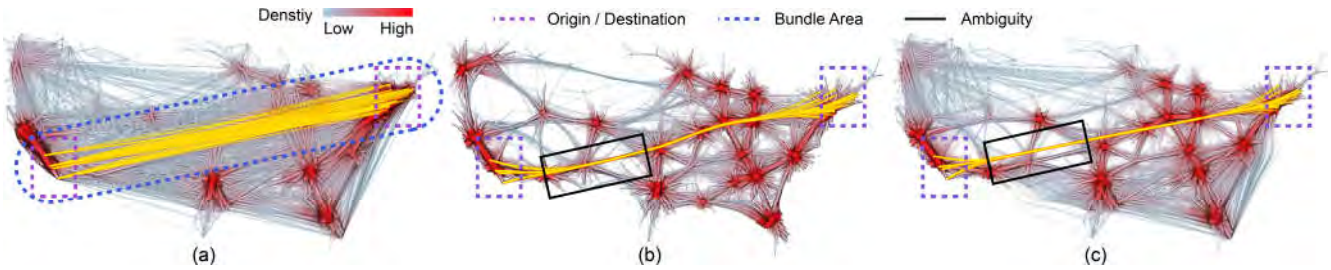

Fig. 14. This example shows the usage of the origin-destination feature. (a) The original US migration dataset with the links in yellow between the selected source and destination nodes, and the bounding box of these links in blue; (b) the bundled result generated by applying the KDEEB algorithm [17] to (a); and (c) further aggregated result by our method, where the ambiguity between edges shown in the black box is resolved.

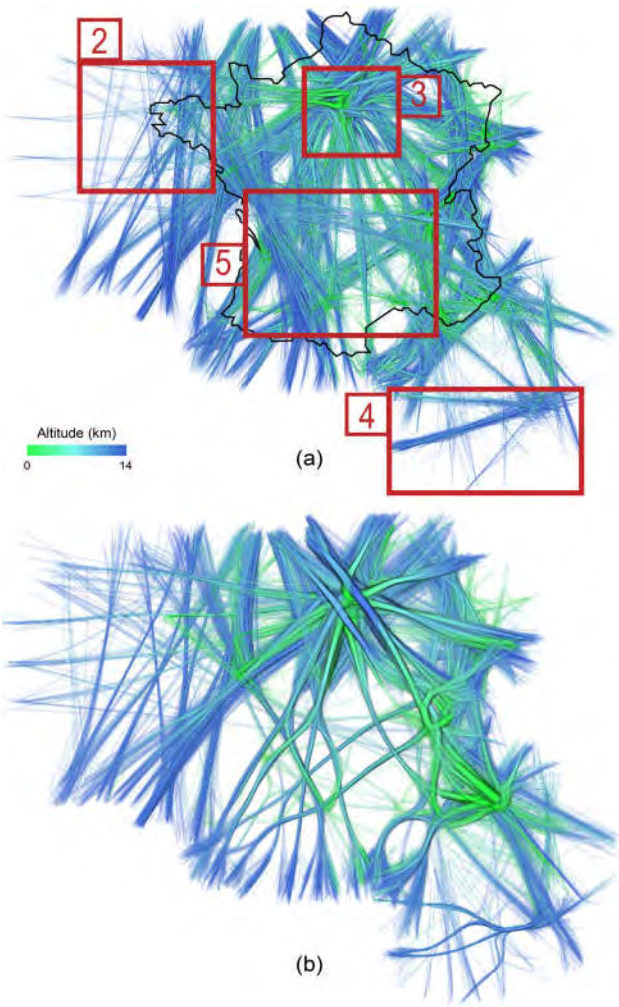

Fig. 13. (a) Original data of the one-day aircraft trajectory record over France. Red boxes show areas, where our method blended different edge bundling results from Fig. 12 to compose the final image shown in (b). (b) Final result produced by our blending method, which takes into account the selected area (red boxes in (a)) using different renderings and edge bundling parameters. Our method insures smooth transitions between areas and also avoids ambiguity.

7.3 Structure-aware Fisheye Views

In the case of general graph exploration, complicated graph structures lead to excessive visual clutter for dense regions. To address this issue, fisheye views [26] have been widely used to show the focal regions with the global context in a single view via space distortion. Although a few fisheye techniques [5, 53] have been proposed to reduce the spatial distortion and improve graph readability, the occlusion between unrelated edges and the regions of interest is still an impediment to the comprehension of local graph structures [51]. Thanks to the blending method, such occlusion can be readily removed by applying the optimization with the readability constraint. As shown in Fig. 3(b), we can first bundle the related edges and re-route them along the outline of the zoomed focal area.

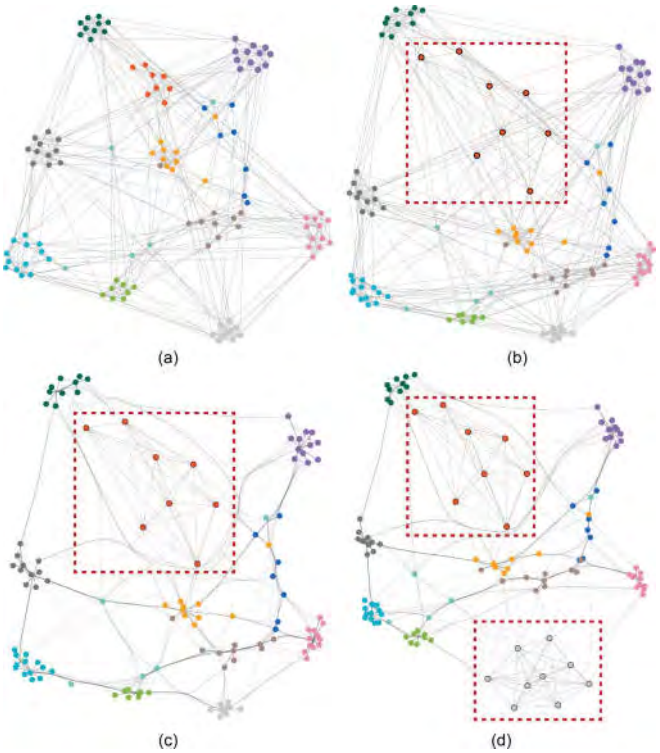

Fig. 15. (a) Input unbundled graph. (b) Cluster fisheye lens zoom-in on the red cluster with black border by (c) bundling the context area while keeping the focus area unbundled. Route the unrelated edges, which go through the focused cluster. (d) Multi-focus fisheye lens on the red and purple clusters, and bundle the associated context area.

Here, we used the FOOTBALL [51] dataset with 115 players (nodes), 613 relations (edges), and 12 clusters to demonstrate the effectiveness of our technique in improving the fisheye view. Fig. 15(a) shows the initial layout generated by Force Atlas2 [20], where the detail of each cluster cannot be clearly shown. To explore the red cluster in the middle of the layout, the cluster fisheye lens is applied, resulting in the layout shown in Fig. 15(b). However, there are many long edges crossing this cluster, hindering users from discerning the connection between nodes of this cluster. Accordingly, we applied the technique that further routes such unrelated edges away from the convex hull of this cluster and obtained the result as shown in Fig. 15(c).

After examining the red cluster, the user would like to further explore

the gray cluster and compare its local structure with the one in the red cluster. Hence, we first applied a polyfocal lens [53] to simultaneously magnify both regions and employed our technique to route the unrelated edges in both focal areas; see the result in Fig. 15(d). We can see that although the gray cluster has a similar number of nodes with the red one, the gray cluster is much denser, which is almost a complete subgraph. This example shows how the proposed technique can help graph exploration with improved graph readability.

8 Discussion

As far as we know, this study provides the first edge bundling composition algorithm taking into account local constraints. This adds flexibility for the user to flexibly produce fine-tuned visual aggregation results that cannot be easily achieved before. Since the algorithm tries to find a suitable transition between two layouts, the user can easily predict the end result. As such, the algorithm does not act as a black box and shows the user expected results as long as the two layouts do not have larger difference. Even if this algorithm remains relativity intuitive, it does not solve the edge bundling parameter complexity. Edge bundling parameters are often obscure and the users often struggle to find suitable edge bundling parameters to produce desirable results. These issues remain pertinent for existing edge bundling techniques [32, 54] in general, and shall be addressed in the future with more intuitive and predictive edge bundling algorithms.

It should be noted that this technique does not provide a novel edge bundling algorithm but rather takes advantage of existing ones. As such, our method is an edge bundling agnostic technique. It can handle any kind of distortion techniques. In the flight dataset use case, we applied the method with common pixel-based edge bundling technique, but other methods could be readily considered. Our method can also take into account any kind of layout algorithms, indeed a use case with a fisheye distortion combined with an edge bundling technique is shown in Fig. 15. In terms of limitations, our method tries to find a suitable optimization to fulfill smoothness, coherency, and ambiguity-free constraints. If there are heavy overlaps among edges in the destination layout, our method may produce cluttered blending results (see Fig. 16). In the future, we plan to further model edge-edge overlap into the readability constraint.

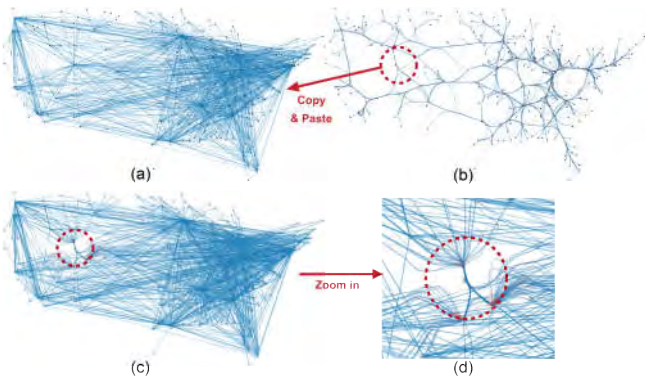

Fig. 16. This figure shows one of our method limitation when blending too divergent graph layouts. (a) the original US Migration dataset; (b) the bundled version of this dataset; (c) our output; and (d) a magnified view of our blending method with strong edge-edge overlap.

9 Conclusion

In this paper, we explored how to leverage existing edge bundling techniques by flexibly aggregating their layout results to produce the final one. Rather than providing a novel edge bundling technique, we proposed to take advantage of existing ones and to provide novel tools to the user to build efficient aggregation results of graphs or trail-sets. Here, we developed an algorithm and a processing pipeline to blend subsets of graph or trail-set layout in a smooth and coherent way. The user can define areas of interest and compose them into a final single view. The algorithm will then process a smooth transition between the blended parts with limited edge distortion. Furthermore, we considered removing ambiguity when edges overlap with nodes and we added three interactive extensions to locally deform, and thus, fine-tune the end result with addition distortion. Finally, we provide a post processing rendering layer to sort the edges and assign edges with a width and a color coding. Overall, the processing pipeline gives the user further flexibility as demonstrated through graph and trail-set examples.

While the pipeline is flexible, some additional features may be considered. Currently, the pipeline only supports the blending of two layers, and some additional work is needed where more than two overlapping layers are to be blended. This technique best fits edge bundling algorithms, since the layout does not differ significantly. We can extend the algorithm to consider the change of end point location to be more generally applicable to any kind of graph layout algorithm. Last, our approach allows users to create a custom edge bundling, but it has a potential for misuse, where users could only show the bundled edges that are supportive of their argument/hypothesis. In the future, we would like to visualize the uncertainty of bundled results by showing which parts have been bundled and how much.

Acknowledgments

This work is supported by the grants of the National Key Research & Development Plan of China (2016YFB1001404), NSFC (61772315, 61861136012), Shenzhen Science and Technology Program (Project no. JCYJ20170413162617606), the Research Grants Council of the Hong Kong Special Administrative Region (Project no. CUHK 14203416), and the French National Agency for Research (Agence Nationale de la Recherche ANR) under the grant ANR-14-CE24-0006-01 project "TERANOVA".

References

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

[2] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum*, 36(1):133–159, 2017. doi: 10.1111/cgf.12791

[3] T. Dang and A. Forbes. Cactustree: A tree drawing approach for hierarchical edge bundling. In *Proc. IEEE Pacific Visualization Symposium*, pp. 210–214, April 2017. doi: 10.1109/PACIFICVIS.2017.8031596

[4] T. Dang, P. Murray, R. Etemadpour, and A. G. Forbes. A user study of techniques for visualizing structure and connectivity in hierarchical datasets. In *VOILA@ISWC*, 2017.

[5] F. Du, N. Cao, Y.-R. Lin, P. Xu, and H. Tong. isphere: Focus+ context sphere visualization for interactive large graph exploration. In *Proc. Conf. on Human Factors in Computing Systems*, pp. 2916–2927, 2017. doi: 10.1145/3025453.3025628

[6] T. Dwyer. Scalable, versatile and simple constrained graph layout. *Computer Graphics Forum*, 28(3):991–998, 2009. doi: 10.1111/j.1467-8659.2009.01449.x

[7] T. Dwyer, Y. Koren, and K. Marriott. IPSep-CoLa: An incremental procedure for separation constraint layout of graphs. *IEEE Trans. Vis. & Comp. Graphics*, 12(5):821–828, 2006. doi: 10.1109/TVCG.2006.156

[8] P. Eades. A heuristic for graph drawing. *Congressus numerantium*, 42:146–160, 1984.

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

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

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

[12] A. Graser, J. Schmidt, F. Roth, and N. Brandle. Untangling origin-destination flows in geographic information systems. *Information Visualization*, 18(1):153–172, 2019. doi: 10.1177/1473871617738122

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

[14] D. Holten and J. J. van Wijk. Force-directed edge bundling for graph visualization. *Computer Graphics Forum*, 28(3):670–677, 2009. doi: 10.1111/j.1467-8659.2009.01450.x

[15] W. Huang, S.-H. Hong, and P. Eades. Effects of crossing angles. In *Proc. IEEE Pacific Visualization Symposium*, pp. 41–46, 2008. doi: 10.1109/pacificvis.2008.4475457

[16] C. Hurter. *Image-Based Visualization: Interactive Multidimensional Data Exploration*. Synthesis Lectures on Visualization. Morgan & Claypool, 2015. doi: 10.2200/S00688ED1V01Y201512VIS006

[17] C. Hurter, O. Ersoy, and A. Telea. Graph Bundling by Kernel Density Estimation. (31):865–874, 2012. doi: 10.1111/j.1467-8659.2012.03079.x

[18] C. Hurter, S. Puechmorel, F. Nicol, and A. Telea. Functional decomposition for bundled simplification of trail sets. *IEEE Trans. Vis. & Comp. Graphics*, 24(1):500–510, Jan 2018. doi: 10.1109/TVCG.2017.2744338

[19] C. Hurter, A. Telea, and O. Ersoy. MoleView: An Attribute and Structure-Based Semantic Lens for Large Element-Based Plots. *IEEE Trans. Vis. & Comp. Graphics*, 17(12):2600–2609, 2011. doi: 10.1109/TVCG.2011.223

[20] M. Jacomy, S. Heymann, T. Venturini, and M. Bastian. Forceatlas2, a continuous graph layout algorithm for handy network visualization. *Medialab center of research*, 560, 2011. doi: 10.1371/journal.pone.0098679

[21] T. J. Jankun-Kelly, T. Dwyer, D. Holten, C. Hurter, M. Nöllenburg, C. Weaver, and K. Xu. *Scalability Considerations for Multivariate Graph Visualization*, pp. 207–235. Springer International Publishing, Cham, 2014. doi: 10.1007/978-3-319-06793-3_10

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

[23] U. Kister, P. Reipschläger, and R. Dachselt. Multilens: Fluent interaction with multi-functional multi-touch lenses for information visualization. In *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces*, ISS '16, pp. 139–148. ACM, New York, NY, USA, 2016. doi: 10.1145/2992154.2992168

[24] J. F. Kruiger, A. Hassoumi, H.-J. Schulz, A. Telea, and C. Hurter. Multidimensional data exploration by explicitly controlled animation. *Informatics*, 4(3), 2017. doi: 10.3390/informatics4030026

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

[26] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *CHI*, vol. 95, pp. 401–408. Citeseer, 1995. doi: 10.1145/223904.223956

[27] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, BELIV '06, pp. 1–5, 2006. doi: 10.1145/1168149.1168168

[28] A. Lhuillier, C. Hurter, and A. Telea. FFTEB: Edge bundling of huge graphs by the Fast Fourier Transform. In *Proc. IEEE Pacific Visualization Symposium*, 2017. doi: 10.1109/pacificvis.2017.8031594

[29] A. Lhuillier, C. Hurter, and A. Telea. State of the art in edge and trail bundling techniques. *Computer Graphics Forum*, 2017. doi: 10.1111/cgf.13213

[30] S. Luo, C. Liu, and K. L. Ma. Ambiguity-free edge-bundling for interactive graph visualization. *IEEE Trans. Vis. & Comp. Graphics*, 18(5):810–821, 2012. doi: 10.1109/tvcg.2011.104

[31] F. J. Newbery. Edge concentration: A method for clustering directed graphs. *SIGSOFT Softw. Eng. Notes*, 14(7):76–85, Oct. 1989. doi: 10.1145/73337.73350

[32] Q. Nguyen, P. Eades, and S.-H. Hong. On the faithfulness of graph visualizations. In *Visualization Symposium (PacificVis), 2013 IEEE Pacific*, pp. 209–216. IEEE, 2013.

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

[34] V. Peysakhovich, C. Hurter, and A. Telea. Attribute-driven edge bundling for general graphs with applications in trail analysis. In *Proc. IEEE Pacific Visualization Symposium*, pp. 39–46, 2015. doi: 10.1109/PACIFICVIS.2015.7156354

[35] H. C. Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing*, 13(5):501–516, 2002. doi: 10.1016/S1045-926X(02)90232-6

[36] H. C. Purchase, C. Pilcher, and B. Plimmer. Graph drawing aesthetics created by users, not algorithms. *IEEE Trans. Vis. & Comp. Graphics*, 18(1):81–92, 2012. doi: 10.1109/TVCG.2010.269

[37] N. H. Riche, T. Dwyer, B. Lee, and S. Carpendale. Exploring the design space of interactive link curvature in network diagrams. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pp. 506–513, 2012. doi: 10.1145/2254556.2254652

[38] H.-J. Schulz and C. Hurter. Grooming the hairball - how to tidy up network visualizations? In *INFOVIS 2013, IEEE Information Visualization Conference*. Atlanta, United States, Oct. 2013. Tutorial.

[39] L. Shao, A. Mahajan, T. Schreck, and D. J. Lehmann. Interactive regression lens for exploring scatter plots. *Computer Graphics Forum*, 36(3):157–166, 2017. doi: 10.1111/cgf.13176

[40] B. Silverman. Density estimation for statistics and data analysis. *Monographs on Statistics and Applied Probability*, 26, 1992.

[41] R. Strzodka and A. Telea. Generalized distance transforms and skeletons in graphics hardware. In *Proceedings of the Sixth Joint Eurographics - IEEE TCVG Conference on Visualization*, VISSYM'04, pp. 221–230. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2004. doi: 10.2312/VisSym/VisSym04/221-230

[42] A. Telea and O. Ersoy. Image-based edge bundles: Simplified visualization of large graphs. In *Proceedings of the 12th Eurographics / IEEE - VGTC Conference on Visualization*, EuroVis'10, pp. 843–852. The Eurographs Association & John Wiley & Sons, Ltd., Chichester, UK, 2010. doi: 10.1111/j.1467-8659.2009.01680.x

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

[44] C. Tominski, S. Gladisch, U. Kister, R. Dachselt, and H. Schumann. A Survey on Interactive Lenses in Visualization. In R. Borgo, R. Maciejewski, and I. Viola, eds., *EuroVis - STARs*, 2014. doi: 10.2312/eurovisstar.20141172

[45] C. Tominski, S. Gladisch, U. Kister, R. Dachselt, and H. Schumann. Interactive lenses for visualization: An extended survey. *Computer Graphics Forum*, 36(6):173–200, 2017. doi: 10.1111/cgf.12871

[46] E. R. Tufte and P. Graves-Morris. *The visual display of quantitative information*, vol. 2. Graphics press Cheshire, CT, 1983.

[47] S. Van den Elzen and J. J. Van Wijk. Multivariate network exploration and presentation: From detail to overview via selections and aggregations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2310–2319, 2014. doi: 10.1109/TVCG.2014.2346441

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

[49] J. Vollmer, R. Mencl, and H. Mueller. Improved laplacian smoothing of noisy surface meshes. *Computer Graphics Forum*, 18(3):131–138, 1999. doi: 10.1111/1467-8659.00334

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

[51] Y. Wang, Q. Shen, D. Archambault, Z. Zhou, M. Zhu, S. Yang, and H. Qu. Ambiguityvis: Visualization of ambiguity in graph layouts. *IEEE Trans. Vis. & Comp. Graphics*, 22(1):359–368, 2016. doi: 10.1109/tvcg.2015.2467691

[52] Y. Wang, Y. Wang, Y. Sun, L. Zhu, K. Lu, C.-W. Fu, M. Sedlmair, O. Deussen, and B. Chen. Revisiting stress majorization as a unified framework for interactive constrained graph visualization. *IEEE Trans. Vis. & Comp. Graphics*, 24(1):489–499, 2018. doi: 10.1109/tvcg.2017.2745919

[53] Y. Wang, Y. Wang, H. Zhang, Y. Sun, C.-W. Fu, M. Sedlmair, B. Chen, and O. Deussen. Structure-aware fisheye views for efficient large graph exploration. *IEEE Trans. Vis. & Comp. Graphics*, 25(1):566–575, 2019. doi: 10.1109/tvcg.2018.2864911

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

[55] D. M. Young. *Iterative solution of large linear systems*. Elsevier, 2014.