# Structure-aware Fisheye Views for Efficient Large Graph Exploration

Yunhai Wang, Yanyan Wang, Haifeng Zhang, Yinqi Sun,
Chi-Wing Fu, Michael Sedlmair, Baoquan Chen and Oliver Deussen

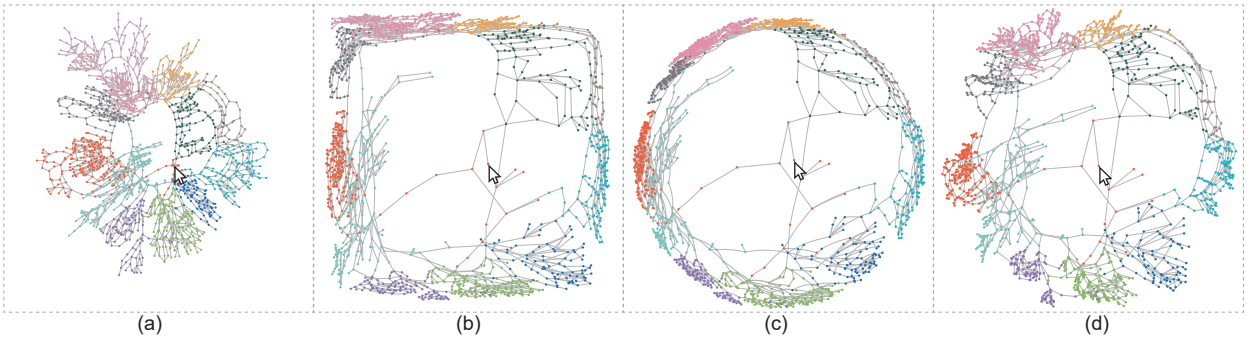


Fig. 1. Magnifying a node-link diagram (a) with 11 clusters around a user-specified location (indicated by the cursor) using different fisheye lenses: (b) graphical fisheye; (c) hyperbolic fisheye; and (d) our structure-aware fisheye, which aims to maintain the shapes of almost all clusters and to minimize their distortions, such as in (b,c).

**Abstract**—Traditional fisheye views for exploring large graphs introduce substantial distortions that often lead to a decreased readability of paths and other interesting structures. To overcome these problems, we propose a framework for *structure-aware* fisheye views. Using edge orientations as constraints for graph layout optimization allows us not only to reduce spatial and temporal distortions during fisheye zooms, but also to improve the readability of the graph structure. Furthermore, the framework enables us to optimize fisheye lenses towards specific tasks and design a family of new lenses: polyfocal, cluster, and path lenses. A GPU implementation lets us process large graphs with up to 15,000 nodes at interactive rates. A comprehensive evaluation, a user study, and two case studies demonstrate that our structure-aware fisheye views improve layout readability and user performance.

**Index Terms**—Graph Visualization, Focus+Context Technique, Structure-aware Zoom, Graph Layout Technique

## 1 Introduction

Graphs are very general representations for describing relationships among entities. They are widely used in most fields of science and model, e.g., social and biological networks, mental processes, road systems, or document and citation networks. Most commonly, they are visualized as node-link diagrams. While many automatic algorithms for creating the layout of such node-link diagrams exist [25], complex graphs with large number of nodes and edges often lead to excessive visual clutter, thereby disguising prominent structures in the graph.

A common approach to mitigate this scalability problem is to use interaction. Pan-and-Zoom, for instance, allows users to explore regions of interest in greater detail, but at the cost of losing the global context of the graph. Hence, pan-and-zoom are often used together with overview+detail techniques [27], where an extra map view is provided for overviewing the entire data. Using such a map view, users can associate and locate the local zoom region in the overall data domain. Although the method is easy to implement and allows users to see the necessary details, using two separated views might break interesting structures, e.g., long paths, clusters, or other mid-scale structures. Furthermore, an additional view consumes precious screen space.

An alternative family of techniques is focus+context methods [33], which aim at showing the focal details with the global context in a single view. This is often achieved by using *fisheye views*, which magnify a local region of higher interest, while compacting other regions of lower importance. Hence, fisheye techniques not only address the issue of losing the spatial continuity but also offer smooth data exploration through pan-and-zoom.

Traditional fisheye methods, however, come with a number of limitations, which restrict their applications to large-scale graph exploration. Geometry-based fisheye techniques, e.g., the graphical fisheye [42] or the hyperbolic browser [37], ignore the structure of a graph while zooming, thereby bringing substantial distortions; see Figures 1(b) and (c) for examples, where the global shape of a graph is heavily distorted. Topological fisheyes [1, 23] seek to overcome this problem by preserving the graph structure based on a hierarchy of coarsened graphs. Such a hierarchy, however, may not be available for general graphs. Furthermore, most existing methods do not attempt to improve the readability [11] of a graph layout during the magnification, e.g., by avoiding node overlaps [14], this would show a graph more clearly to the user. EdgeLens [55] and its successors [36, 48] improve the readability of edges around the focal points by pushing away their neighbor edges. They do not support magnification, though, and thus the improvement on readability is limited. Lastly, methods like hyperbolic views and

- *Y.H. Wang, Y.Y. Wang, H. Zhang and Y. Sun are with Shandong University. Email: {cloudseawang, yanyanwang93, hiphone.zhang,sunyinqi0508}@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. E-mail: cwfu@cse.cuhk.edu.hk.*
- *M. Sedlmair is with VISUS, University of Stuttgart, Germany. E-mail: michael.sedlmair@visus.uni-stuttgart.de.*
- *O. Deussen is with Konstanz University, Germany and Shenzhen VisuCA Key Lab, SIAT, China. E-mail: oliver.deussen@uni-konstanz.de.*
- *Y.H. Wang and Y.Y. Wang are joint first authors and H. Zhang and Y. Sun contribute equally to this work.*

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

iSphere [10, 37] render links as curves. This, however, might hamper graph exploration, especially for tasks such as path tracing [10, 56].

In this paper, we present a unified optimization framework to produce *structure-aware* fisheye views for efficient exploration of large graphs. The framework is based on three requirements for good focus+context visualization design, identified by Cohé et al. [8]: (i) minimizing the distortion of the structure of a diagram; (ii) improving the readability of nodes of interest; and (iii) maintaining a coherent layout during the interaction. These requirements are also supported by recent empirical evaluations of graph exploration techniques [10, 15, 56]. As far as we know, these requirements, so far, have not been adopted in any fisheye technique for exploring node-link diagrams, though some are met in the force-based overview+detail interface by Dwyer et al. [15]. SchemeLens [8] also fits these requirements well, but only deals with vector-based network diagrams (with only horizontal and vertical lines) rather than general node-link diagrams, and provides only a simple, axis-aligned structural focus.

Our key approach is to formulate the fisheye zoom as an optimization problem, with a goal of delivering an *interactive*, *smooth*, and *structure-aware* zoom by preserving the *edge orientations* in the graph layout. By including constraints for edge orientations into the optimization, we are able to model various types of layout constraints for the zoom, including structural and temporal constraints, for an efficient layout production. With our new formulation, we can effectively minimize the distortions of local structures and better maintain the spatial continuity; see Figure 1(d) for one of our results. Moreover, we can also improve the readability of a graph layout by imposing aesthetic constraints such as minimizing the node overlap, similar to constraint-based layouts [14, 54]. In addition, our new formulation allows users to interactively specify structures to be preserved during the fisheye interaction, e.g., sketching a circle or a hierarchy, while maintaining the overview of the graph, which itself could be large and highly complex.

On the other hand, we are able to design various task-oriented *fisheye lenses* in our framework according to a task taxonomy [34] for graph visualization: (i) a *polyfocal* lens for highlighting structures around multiple focus points, (ii) a *cluster* lens for maintaining a substructure, and (iii) a *path* lens for focusing on and preserving a user-defined path. These lenses together can be used for exploring multiple structures of interest at the same time.

We develop an efficient GPU-based implementation, enabling an interactive exploration of large graphs with up to 15K nodes and various structures to be preserved. We quantitatively compare our method to previous approaches by measuring the change of structures by using a number of evaluations and studies to demonstrate the effectiveness and applicability of our fisheye lens for exploring graphs.

In summary, the main contributions of this paper are:

- we formulate a unified framework to generate structure-aware fisheye views for exploring large graphs;
- we devise a family of task-oriented fisheye lenses for different graph visualization tasks; and
- we develop an interactive GPU-based implementation, and demonstrate the effectiveness of our method through a comprehensive evaluation, a user study, and two case studies.

## 2 Related Work

### 2.1 Graph Exploration Techniques

Various interaction techniques [53] have been designed to support essential tasks in graph exploration. Lee et al. [34] summarize a list of tasks, among which attribute-based and topology-based tasks are two major categories. Attribute-based tasks refer to finding nodes or links with certain labels, and are often accomplished by extracting and visualizing the related sub-graphs based on a search query [5, 40, 50]. Topology-based tasks include examining the adjacency between nodes and finding the shortest path between them, as well as identifying clusters, which are essential for navigating graph structures. In this paper, we focus on interaction techniques for topology-based tasks, which can be further categorized into four classes [26]: pan-and-zoom, fisheye views, semantic zooming, and dynamic layouts.

**Pan-and-zoom** is the most commonly used interaction technique, it allows users to select and zoom into a region of interest. Among the existing zoomable interfaces [39, 51], several are used for graph exploration [47, 52]. This interaction approach, however, has some inherent limitations such as insufficient navigation patterns and disorientation of the user [27]. Hence, zoomable interfaces are often used with an additional overview that shows the entire data.

For graph exploration, such interfaces [17, 43, 49] typically have two separate views: an overview window and a detailed view for small sub-graphs of interest. In most cases, node-link diagrams are shown in both views using the same layout technique, thus the readability of the graph structures in both views are almost the same. To improve the readability of the detailed view, Dwyer et al. [15] generate high-quality layouts by incorporating graph drawing aesthetics. Likewise, our approach attempts to improve the readability of the focal regions using aesthetic constraints in the form of edge orientations. Moreover, our method can support interactive exploration of large graphs with 15K nodes, whereas the best current state-of-the-art system can only handle graphs of some hundred nodes due to the need for a complex optimization [16].

**Fisheye views** [20] are well-known examples of focus+context techniques that seamlessly incorporate detailed focus regions within a global context via space distortion. Several graph-specific fisheye techniques have been proposed, such as graphical fisheye [42], hyperbolic display [37], and iSphere [10]. A graphical fisheye directly transforms the graph's 2D layout, while the latter two map the 2D layout into a non-Euclidean geometry space (hyperbolic surface or Riemann Sphere) and then project it back to the original 2D space. Although these methods distort the space with different geometric transformations, both distort nodes simply based on the node's distance to the focus. Doing so, the structure defined by the graph's edges is completely ignored, and thus, large distortions to the shape of the graph might occur. For simplicity, we refer to these methods as *geometric fisheyes*.

To preserve a given structure, topological fisheye techniques [23] employ a pre-computed hierarchy of coarsened graphs to guide distortion. Constructing such a hierarchy, however, is nontrivial for general graphs, where structure preserving operations are required. In contrast, our approach provides an efficient formulation based on edge orientations of the input layout to maintain graph structures; by this new means we are able to preserve structures with better quality and performance than previous methods. It should be noted that Ti et al. [46] also constrain the geometric magnification with the constraints of edge orientations. However, the method treats the horizontal and vertical axes separately in the maps, whereas our method can directly handle arbitrary directions in general graphs and formulates the optimization as a linear system.

**Semantic zooming** [21] is commonly used for exploring hierarchical graphs [1, 18], where the input layout is hierarchically clustered into different levels of abstraction. By default, it shows the coarsest level; then users can zoom in and explore next levels at will. In other words, the navigation is restricted by the displayed size of the graph. To address the issue, topological fisheye views [23] leverage *hybrid graphs*, generated by merging detailed regions from the original graph with other regions from coarser representations, to which a geometric fisheye view can be applied. Our approach can be seen as a combination of semantic zoom and geometric fisheye, where the structure is preserved but the distortion is based on a geometric transformation.

**Dynamic layout** aims at improving the readability of regions of interest by optimizing the graph layout. EdgeLens [55] interactively displaces the edges away from the focal point in order to reveal the region around the focal nodes and edges. LocalEdge lens and BringNeighbors lens [48] are variations of EdgeLens that filter edges between nodes within a focus region and bring neighboring nodes closer to the nodes of interest. MoleView lens [29] introduces continuous bundling to show the structure of edges of interest at different levels. Our approach is to interactively improve the graph's readability through an optimization, where aesthetic constraints can also be integrated efficiently.

### 2.2 High-quality Graph Layout Techniques

A wide variety of automatic graph layout algorithms have been developed for graph visualization [45], among which stress-based methods [32] are very popular. These methods produce good layouts by adjusting node positions to optimize the distances between nodes. Only optimizing the distance constraints, however, does not allow for effects such as avoiding edge overlaps or preserving structures.

To address these issues, constrained graph layout methods [6] incorporate geometric constraints into the optimization. Early attempts focus on directional constraints [24,30,44]; however, the corresponding optimization problems are usually NP-hard. Thus, existing heuristic solving procedures might not produce satisfying layouts, especially for large graphs. Dwyer et al. [13] propose Dig-CoLa that efficiently solves the optimization of constraint layouts by integrating constraints into stress majorization [22]. Later, they extend stress majorization to satisfy separation constraints [14] as well as non-linear constraints [12]. More recently, Wang et al. [54] reformulate stress majorization in a vector form, so that various constraint types can be defined by edge orientations and used in the optimization. We follow this stream of research but formulate a new optimization model with structural, readability, and temporal constraints for producing the fisheye views, and solve for these constraints with quadratic time complexity.

## 3 Background

Given a graph $\{\mathbf{V},\mathbf{E}\}$ with nodes $\mathbf{V}$ and edges $\mathbf{E}$, such that each node has a position $\mathbf{x}_i \in \mathbf{R}^2$ and $\mathbf{X} = \{\mathbf{x}_1,\cdots,\mathbf{x}_n\}$ denotes the set of all positions. Using these notations, we first briefly describe some relevant state-of-the-art geometric fisheye methods and then introduce the edge-vector formulation of stress majorization [54], which inspired us to develop our framework for structure-aware fisheye views.

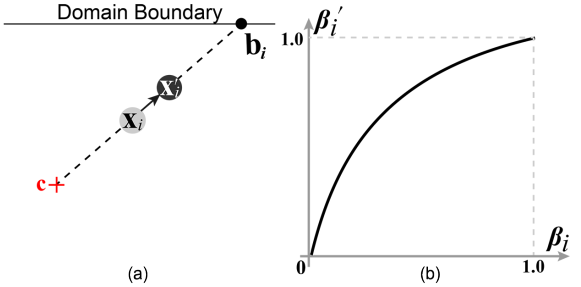


Fig. 2. Graphical fisheye: (a) each node $\mathbf{x}_i$ is moved away from the focal point $\mathbf{c}$ toward its boundary point $\mathbf{b}_i$ along a line that joins them; (b) distortion function ($m = 3$) for the distance ratio $\beta_i$; see Eq. (1) for details.

### 3.1 Geometric Fisheye Views

Common geometric fisheye methods include the graphical fisheye [42], hyperbolic fisheye [37], and iSphere [10]. The graphical fisheye directly applies the geometric transformation in the planar domain, while the hyperbolic fisheye and iSphere first map the graph layout to a hyperboloid and Riemann sphere, respectively, and then project the result back onto the Poincaré disk and a plane, respectively. Below, we briefly review the graphical fisheye, which is to be extended to form the task-driven lens (Section 5) for graph exploration.

**Graphical fisheye.** Given a graph layout $\mathbf{X}$ and a focal point $\mathbf{c}$, a graphical fisheye magnifies the graph by displacing each $\mathbf{x}_i$ away from $\mathbf{c}$. The method first locates point $\mathbf{b}_i$ on the domain boundary by extending a line from $\mathbf{c}$ through $\mathbf{x}_i$ (see Figure 2(a)) and then displaces $\mathbf{x}_i$ to $\mathbf{x}_i'$ along the line toward $\mathbf{b}_i$ by the following equation:

$$\mathbf{x}_i' = \mathbf{c} + (\mathbf{b}_i - \mathbf{c})\beta_i', \text{ where } \beta_i' = \frac{(m+1)\beta_i}{m\beta_i + 1},\ \beta_i = \frac{||\mathbf{x}_i - \mathbf{c}||}{||\mathbf{b}_i - \mathbf{c}||}, \quad (1)$$

$m \geq 0$ is the magnification factor in the fisheye zoom. Note that $\beta_i$ is a distance ratio to be increased nonlinearly to $\beta_i'$ (see Figure 2(b)); then, the method takes $\beta_i'$ to create the new position $\mathbf{x}_i'$.

All these three fisheye views apply the geometric transformation solely to the node positions, while completely ignoring the edge connections. Hence, the magnification could heavily distort the graph structure; see again Figures 1 (b) and (c). Through our new formulation, we can effectively magnify a graph layout with a fisheye zoom, while maintaining its structures in the context; see Figure 1(d).

### 3.2 Edge Vector-based Stress Majorization

Recently, Wang et al. [54] reformulated stress majorization as an optimization in vector form that allows users to explicitly control not only the *edge lengths* but also the *edge orientations* in graph layouts. The optimization is formulated in the following form:

$$\arg\min_{\mathbf{X}} \sum_{i<j} w_{ij}(\mathbf{x}_i - \mathbf{x}_j - \mathbf{e}_{ij}d_{ij})^2\ , \quad (2)$$

where $w_{ij}$ is a normalization weight, and $\mathbf{e}_{ij}$ and $d_{ij}$ are the target orientation (expressed as a unit vector) and length, respectively, for the edge between nodes $\mathbf{x}_i$ and $\mathbf{x}_j$. Using this model, users can specify various types of constraints in terms of edge lengths and orientations in order to alter the graph layout; please refer to [54] for details.

Inspired by the reformulated model, we aim at producing structure-aware and smooth fisheye views at interactive rates by modifying edge orientations and lengths and by formulating various types of constraints (structure-based, readability-based, and temporal) for supporting the fisheye zooms (Section 4). In addition, we also design a family of special fisheye lenses for different exploration tasks (Section 5).

## 4 Structure-Aware Fisheye

Based on the three requirements described in the introduction (structure preservation, readability, and layout coherency), we formulate our structure-aware fisheye lens as an optimization problem to solve for $\mathbf{Z}^t$ (which is the set of node positions in the $t$-th iteration, so $\mathbf{Z}^0 = \mathbf{X}$) using the following three-fold objective:

$$\begin{aligned}\arg\min_{\mathbf{Z}^t} &\sum_{(i,j)\in\mathbf{E}} \omega_{ij}^s ||\mathbf{z}_i^t - \mathbf{z}_j^t - \mathbf{e}_{ij}^s d_{ij}^s||^2 \\ &+ \sum_{(i,j)\in\Omega} \omega_{ij}^r ||\mathbf{z}_i^t - \mathbf{z}_j^t - \mathbf{e}_{ij}^r d_{ij}^r||^2 \\ &+ \sum_{i\in\{1..n\}} \omega_i^t ||\mathbf{z}_i^t - \mathbf{z}_i^{t-1}||^2\ , \end{aligned} \quad (3)$$

where $\omega_{ij}^s$, $\omega_{ij}^r$ and $\omega_i^t$ are the normalization weights (default value is one), $\Omega$ is the set of edges inside the focal area around $\mathbf{c}$. The parameters $\mathbf{e}_{ij}^s$ and $d_{ij}^s$ provide the structural constraints, $\mathbf{e}_{ij}^r$ and $d_{ij}^r$ provide the readability constraints, and $\mathbf{z}_i^t \in \mathbf{Z}^t$ denotes the position of the $i$-th node in the $t$-th iteration during the fisheye zoom. Our optimization model consists of three terms (see Eq. (3)), each corresponding to one of the three constraints (i.e., structure preservation, readability, and layout coherency): the first term aims to maintain the structures by preserving the edge orientations and lengths for all the edges $\mathbf{E}$ of the graph; the second term implements the readability criterion for edges in $\Omega$; and the last term enforces the temporal coherency, where the nodes are desired to move stably over the iterations during the zoom.

Please note that the set $\Omega$ is not just a subset of $\mathbf{E}$, but also contains additional, invisible edges between all the node pairs of the focal area. We need this strategy to enhance the readability, since we need to push nodes away from one another, including also the nodes that are not explicitly connected in the graph. For more details about these virtual edges and their handling, please refer to [14]. The typical focal area for a fisheye is defined as a circular region around $\mathbf{c}$ with a radius of 20% of the screen size. Except for the constraint of temporal coherence, both the definitions of structure-based constraints and readability constraints are based on the geometric magnification; see Sections 4.1 and 4.2.

### 4.1 Structure-based Constraints

Unlike previous works on graph exploration, the preservation of structure is a tricky part of our problem, since a fisheye lens distorts the

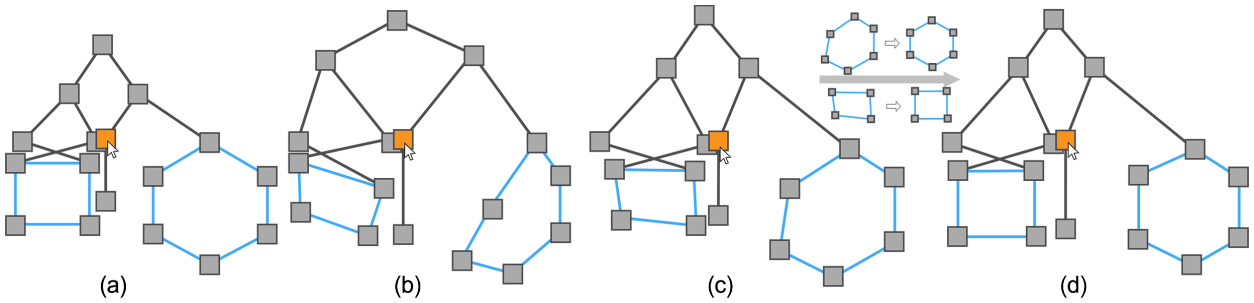


Fig. 3. Structure-based constraints: (a) input layout; (b) result produced by the graphical fisheye; (c) result produced by our method using only the edge orientation constraints; and (d) result produced by our method further using the shape constraints with the edge orientation constraints.

layout non-linearly. To magnify a layout, we have to vary the edge lengths to a large extent and to push the spatial context towards the domain boundary; see Figures 1(a) and (d). Hence, we cannot simply force the edge lengths and orientations in the target layout $\mathbf{Z}^t$ to follow the original layout $\mathbf{X}$ for preserving its structures. Instead, our idea is to encourage the edge orientations in $\mathbf{Z}^t$ to follow the original layout $\mathbf{X}$ and the edge lengths in $\mathbf{Z}^t$ to follow the target layout $\mathbf{X}'$ estimated by the fisheye, so that the edge lengths in $\mathbf{Z}^t$ can be adjusted to accommodate the zoom. Hence, we set $\mathbf{e}^s_{ij}$ and $d^s_{ij}$ in Eq. (3) as follows:

$$\mathbf{e}^s_{ij} = \frac{\mathbf{x}_i - \mathbf{x}_j}{||\mathbf{x}_i - \mathbf{x}_j||} \text{ and } d^s_{ij} = ||\mathbf{x}'_i - \mathbf{x}'_j|| \,. \tag{4}$$

**Shape constraints.** Using the above constraints allows us to produce a fisheye view with a better preservation of its structure. However, these constraints are only local in their nature, thus larger-scale structures may still be distorted; compare the blue loops in Figures 3 (a) to (c).

To enhance the preservation of such structures, we let users provide a connected set of edges, say $\mathbf{E}_s$, in the graph (e.g., one of the blue loops in Figure 3(c)) for us to collectively estimate better $d^s_{ij}$ for the edges. We observe that during a zoom some edges in larger-scale structures could become relatively longer while others could become relatively shorter. Hence, if we can balance the changes of the edge lengths from the original layout $\mathbf{X}$ to the target layout $\mathbf{Z}^t$, the edge orientation $\mathbf{e}^s_{ij}$ should be able to preserve such structures through the optimization.

Denoting $d_{i,j}$ as the length of edge $(i,j)$ in the original layout $\mathbf{X}$ and $d'_{i,j}$ as the length of the same edge in $\mathbf{X}'$, we first look for the average length distortion ratio $\rho$ through the following minimization:

$$\min_{\rho} \sum_{(i,j)\in\mathbf{E}_s} \left(\rho d_{i,j} - d'_{i,j}\right)^2 \,, \tag{5}$$

which can be solved analytically by:

$$\rho = \sum_{(i,j)\in\mathbf{E}_s} \frac{d_{i,j}}{\sum_{(i,j)\in\mathbf{E}_s} d_{i,j}} \frac{\sum_{(i,j)\in\mathbf{E}_s} d_{i,j} d'_{i,j}}{\sum_{(i,j)\in\mathbf{E}_s} d^2_{i,j}} \,. \tag{6}$$

This allows us to define the edge length, i.e., $d^s_{i,j}$, by using $\rho$ and $d_{i,j}$, instead of setting it using Eq. (4):

$$d^s_{i,j} = \rho\, d_{i,j}. \tag{7}$$

In addition, we also increase $\omega^s_{ij}$ (empirically as 10) for the edges in $\mathbf{E}_s$ to strengthen the structure preservation; see Figure 3(d) for a result.

### 4.2 Readability-based Constraints

A few graph drawing readability metrics have been developed and some have been incorporated to create graph layouts [11, 12]. However, none of them has been integrated into the fisheye views, where a clear graph structure is expected in the region of interest. In the context of this work, we consider two kinds of readability constraints:

**Non-overlapping Nodes.** Rather than zero-sized points, nodes in graph drawings are typically represented by dots or little squares with a given size, some nodes may even carry text labels; see Figure 12 for an example. Hence, we should consider node sizes and avoid node overlap, particularly in the focal area.

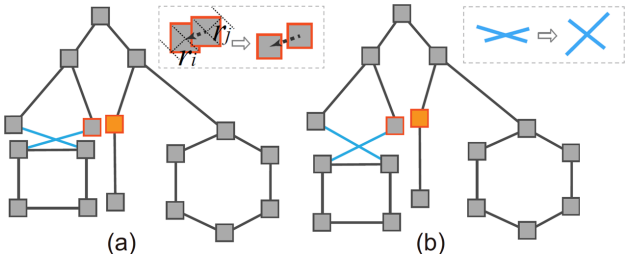


Fig. 4. Readability-based constraints: (a) result produced after pushing the two overlapped nodes in Figure 3(d) away from each other; and (b) result produced after further maximizing the edge crossing angle.

Concerning this, we detect overlapping node pairs (say $\mathbf{x}_i$ and $\mathbf{x}_j$) in the focal area after we applied the structure-based constraints to the layout (see Figure 3(d)) and set an edge length constraint between these node pairs using the readability term in Eq. (3):

$$d^r_{ij} = r_i + r_j + \varepsilon \text{ and } \mathbf{e}^r_{ij} = \mathbf{e}^s_{ij} \,, \tag{8}$$

where $r_i$ denotes the radius of the bounding circle of node $i$ in screen space and $\varepsilon$ is a node separation parameter heuristically set as 1% of the screen size. See Figure 4(a) for a result produced after imposing the node-overlapping constraint to the graph shown in Figure 3(d).

If no visible edge between $\mathbf{x}_i$ and $\mathbf{x}_j$ exists in the graph, we add a virtual edge between the node pair and set its length according to Eq. (8). This is done for all overlapping node pairs in the layout.

**Maximizing edge crossing angle.** Purchase's seminal graph readability study [41] suggested that edge crossings have great impact on human's understanding of graphs. However, completely resolving edge crossings is impossible [30], especially for large graphs. On the other hand, a recent study [28] showed that maximizing the crossing angle can improve the performance of path tracing tasks. Following this finding, we maximize edge crossing angles by the following procedure.

Again, we detect edge crossings inside the focal area (say edge $(i,j)$ and edge $(l,k)$) after we applied all the structure-based constraints to the layout (see Figure 3(d)) and set the directions for $(i,j)$ and $(l,k)$ using the readability term in Eq. (3):

$$\mathbf{e}^r_{ij} = \mathbf{e}^s_{ij} \oplus \frac{\pi - \alpha}{2} \text{ and } \mathbf{e}^r_{lk} = \mathbf{e}^s_{lk} \ominus \frac{\pi - \alpha}{2} \,, \tag{9}$$

where $\alpha$ is the angle between $\mathbf{e}^r_{ij}$ and $\mathbf{e}^r_{lk}$, while $\oplus$ denotes a clockwise and $\ominus$ a counter-clockwise rotation. The edge lengths in this case will not be altered.

### 4.3 Solving the Optimization

To obtain $\mathbf{Z}^t$, we differentiate Eq. (3) with respect to $\mathbf{Z}^t$ and set the derivative to zero, thereby resulting a linear system. We solve this linear system by using a conjugate gradient solver that iteratively finds the

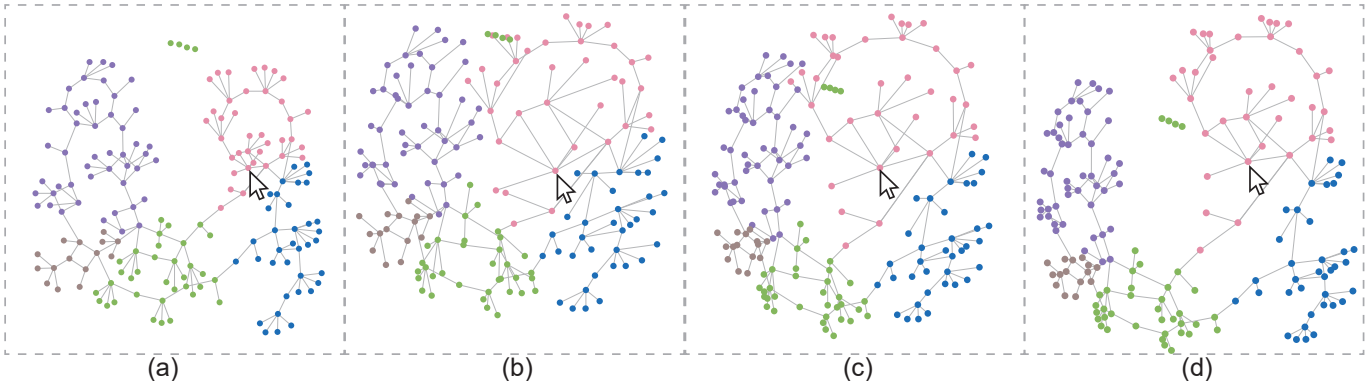


Fig. 5. Convergence of our method, where we perform our fisheye zoom at the focal point marked by a cursor using our structure-based, readability-based, and temporal constraints. (a) Input layout, and (b)-(d) results after 10, 20, and 30 iterations, respectively.

solution. Figure 5 shows the fast convergence of this process, where the result of the 20-th iteration already satisfies most of the constraints. To allow an interactive exploration, we adapted the GPU solver in [54] to incorporate the different constraints in Eq. (3). This enables us to handle large graphs with up to 15K nodes.

## 5 Task-driven Fisheye Lenses

Existing fisheye views for graph exploration mostly support only a single focus, which is efficient for examining the local neighborhood around a focal node. However, this interaction model is too restrictive and fails to support general graph exploration tasks, e.g., exploring the common neighbors of two given nodes, following a path of interest, and exploring clusters. Such tasks involve multiple focal nodes and edges. To this end, we propose a family of task-driven fisheye lenses to support a wider variety of graph exploration tasks. Below, we present the lenses at a high level, and provide their implementation details in the supplemental material.

**Polyfocal Lens.** To generalize the single focus fisheye from Section 4 to support multiple focal centers, we apply the polyfocal projection [31] to the current layout and then define the edge direction and length information in terms of Eq. (4). Figures 11(b) and (c) compare the results generated by the original polyfocal lens and our polyfocal lens, where we can see that our result better preserves the cluster structure.

**Cluster Lens.** A desired lens for exploring a cluster of interest should magnify the cluster, while providing the context of the entire graph. Previous fisheye views are based on one or multiple focal centers. Applying them, however, does not guarantee that all the nodes in the cluster are properly magnified. Therefore, we develop the cluster lens to allow users to specify a convex focal area and then magnify the region (and the cluster) linearly in the fisheye view, while compressing the outside context using the fisheye distortion model.

**Path Lens.** Path exploration tasks, such as exploring a path's neighboring nodes and checking their degrees, require a magnification subject to a path of interest. Yet, we are not aware of any fisheye technique that can support such nontrivial zooms. For this reason, we formulate a path lens in our framework for supporting such tasks.

To create a path lens, a user simply picks two nodes in the graph. Our method then automatically finds and locates the shortest path that connects the two nodes, and defines the focal area around the path. Note that the focal area has a radius of $\sigma$ measured from the path. We empirically set $\sigma$ as $\sqrt{m}/28$ times the screen size, so that the focal area can adaptively increase with the magnification factor $m$.

## 6 Results and Evaluation

We implemented our method in C++ and tested it on a computer with an Intel Core i7 processor with 16GB memory. Moreover, we developed a GPU implementation in CUDA that runs on an NVidia GTX1080 graphics card with 8GB video memory. We performed three experiments to evaluate our structure-aware fisheye views. First, we quantitatively compared it with the state-of-the-art fisheye views (Section 6.1) in terms of structure preservation and readability. Second, we conducted a lab study to compare our method with other state-of-the-art fisheye views on the task of path tracing (Section 6.2). Last, we qualitatively demonstrated its usefulness by two case studies with real-world datasets and by combining the use of different lenses (Section 6.3).

### 6.1 Quantitative Comparison

We compare our approach with three state-of-the-art graph fisheye views: graphical fisheye (GF) [42], hyperbolic fisheye (HF) [37], and iSphere [10]. To illustrate how our approach overcomes their shortcomings, we consider three factors: (i) edge orientation preservation, (ii) node overlapping, and (iii) shape preservation. For a fair comparison, our method only enforces the edge orientation and node non-overlapping constraints. Since the original hyperbolic fisheye views are not developed for exploring general 2D node-link diagrams, we follow the implementation of Du et al. [10], which achieves the pan-and-zoom based on a Möbius transformation [4] and uses the Poincaré metric [3]. Since iSphere projects the layout on a sphere, some of the nodes are lost after the magnification and we cannot obtain all the node and edge information. Hence, we can only compare iSphere on shape preservation but not on the other two factors in the quantitative comparison. However, we may still apply it with our method to produce layouts.

Therefore, taking GF, HF and iSphere as the geometry transformation to find the target layout $\mathbf{X}'$ (see Section 4), there are six methods altogether: GF, HF and iSphere, as well as Ours+GF, Ours+HF, and Ours+iSphere. Note that GF, HF and iSphere are all based on analytical transformations, whereas our methods (the last three combinations) require solving an optimization; see Eq. (3).

We employed five real datasets with varying numbers of nodes and edges; see Table 1 for a summary. For each dataset, we randomly selected 100 focus centers (see the examples in Figure 1) and 20 random magnification factors ranged from 0.1 to 20 for each focus center. To avoid bias due to randomness, we ran the three methods, i.e., Ours+GF, Ours+HF, and Ours+iSphere, 100 times. We found that the running time is almost the same for different focus centers configured with the same magnification factor. The last column in Table 1 reports the range of computing time for solving the optimization (GPU implementation), showing that our fisheye views can be rendered interactively.

Table 1. Datasets used in our quantitative comparison experiment and the range of computing time on each dataset.

| Dataset | #Nodes | #Edges | Time in sec. |
|---|---|---|---|
| SANDI_AUTHORS | 90 | 125 | 0.03± 0.002 |
| BCSPWR1454 | 1454 | 1923 | 0.05± 0.002 |
| FACEBOOK4039 | 4039 | 88234 | 0.14 ± 0.001 |
| BCSPWR10 | 5300 | 8271 | 0.24 ± 0.001 |
| PSSE1 | 14318 | 57366 | 0.91 ± 0.01 |

**Preservation of edge orientations.** To specifically measure to what degree the various methods preserve the edge orientations, we define a

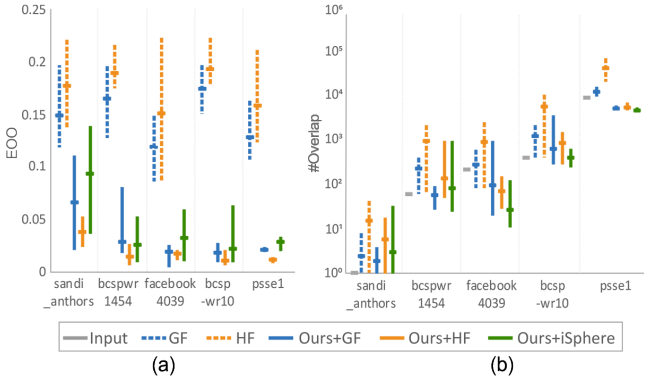


Fig. 6. Comparison of layouts generated by our method and existing fisheye views in terms of (a) edge orientation offset (EOO) and (b) number of overlapped node pairs.

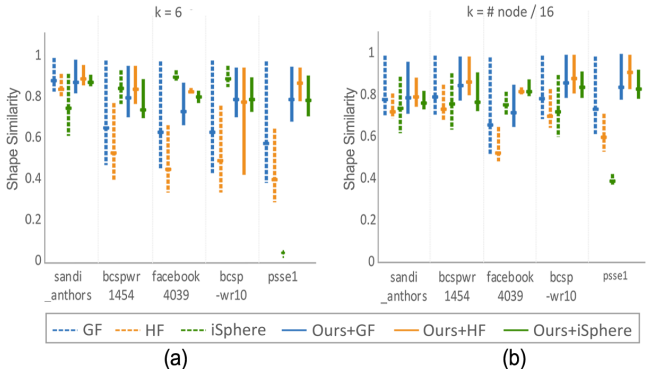


Fig. 7. Comparison of layouts generated by our method and existing fisheye methods in terms of the $k$-nearest neighbor graph based shape similarity. A high value indicates better preservation on the shape structure. We also try the experiment with two different $k$ values.

metric called the *Edge Orientation Offset (EOO)*:

$$\mathrm{EOO}(\mathbf{Z}) = 1 - \frac{1}{|E|}\sum_{(i,j)\in E} \| < \frac{\mathbf{x}_i - \mathbf{x}_j}{\| \mathbf{x}_i - \mathbf{x}_j \|}, \frac{\mathbf{z}_i - \mathbf{z}_j}{\| \mathbf{z}_i - \mathbf{z}_j \|} > \|, \qquad (10)$$

which measures the sum of absolute inner products between the edge vectors in the original layout and the magnified layout (1 - cosine similarity). A small $EOO(\mathbf{Z})$ indicates that most edge orientations are preserved, while a larger value indicates larger changes in the graph structure. Compared with the objective function (Eq. (3)), EOO only measures the changes in edge directions without also involving the edge lengths, so it focuses on revealing the degree of structure preservation. We also do not use the objective function (Eq. (3)) here; as it is used in the design of our methods and hence might bias the study.

Figure 6(a) shows the EOO values for the layouts of five different datasets generated by GF, HF and three versions of our method. All three versions perform significantly better than GF and HF and produce low $EOO(\mathbf{Z})$, i.e., <0.07 for the five datasets, except SANDI_AUTHORS, whose initial layouts already have heavy visual clutter. Among the three versions of our method, Ours+HF is the best, Ours+iSphere is the worst, and Ours+GF is in-between the two.

**Node overlapping.** To study how our node non-overlapping constraint improves the readability, we count the number of overlapped node pairs for the input layout and all magnified layouts. Figure 6(b) shows the results, where the counts of our methods are mostly lower than those by HF and GF. For the first three datasets, the number of overlapped node pairs in the best version of our method is even close to the ones in the input layout, indicating that our fisheye not only magnifies the structure of interest but also improves the readability of the results. Figure 1 shows the BCSPWR1454 dataset, where the number of overlapped node pairs generated by GF and Ours+GF are 398 and 128, respectively.

**Shape preservation.** To measure how the shape structure is preserved by our methods, we use the shape-based metrics proposed by Eades et al. [19] to compare the shape changes before and after the zoom. Given a graph of nodes, these metrics compute a so-called shape graph that consists of the $k$-nearest neighbors for each node. The shape graphs of the input graph and magnified graph are computed and then compared using the mean Jaccard similarity between the two shape graphs, where a larger value indicates better preservation on the shape structure. Note that for iSphere, we can only compare with it in 2D by considering the xy coordinates of the nodes in its generated layouts.

Figure 7 shows the result using two different $k$ values. We can see that our methods preserve the global shape better than the other methods, and iSphere works well for the BCSPWR1454 and BCSPWR10 datasets, especially for small $k$. After carefully looking at the iSphere results, we found that iSphere can better preserve the local shape than the global shape, while our methods work well in both aspects.

Figure 8 shows a comparison of the different approaches, where the input layout is generated by integrating a few shape constraints and cluster non-overlap constraints into the stress model [54]. In the upper row, we show the input layout (a), the application of a graphical fisheye (b), a hyperbolic fisheye (c) and an iSphere (d). In the bottom row, we show a linear magnification (e) and then the results of our approach, where a graphical fisheye (f), a hyperbolic fisheye (g), and an iSphere (h) are used to compute the target layout in our method, respectively. We can see that our results well preserve the shapes of the square, the pentagon and the right angles with less overlaps, while Ours+HF performs the worst among the three versions of our method.

## 6.2 Lab Study

The aim of the lab study is to learn if our structure-aware fisheye views would improve the efficiency of graph exploration. We thus set up a human-subject experiment in which the following five methods are considered: Pan-and-Zoom (P&Z), Graphical Fisheye (GF), Hyperbolic Fisheye (HF), iSphere, and one version of our method. Since the quality of Ours+GF is between the other two versions in the quantitative study (Section 6.1), we choose Ours+GF in the lab study. Further, we do not test our task-driven lens, because traditional methods neither inherently support specific graph exploration tasks nor consider the readability-based constraints. Hence, we only impose the edge orientation constraints in our method for a fair comparison.

### 6.2.1 Experiment

**Task.** Du et al. [10] conducted a comprehensive evaluation of three state-of-the-art techniques (Pan-and-Zoom, Hyperbolic and iSphere) with three tasks that focused on exploring nodes, links, and paths. Since our fisheye views explicitly attempt to preserve the edge orientations, they are supposed to perform better for exploring nodes and links, as verified by our pilot studies. In contrast, tracing a path between nodes raises the necessity to explore a larger context and it is uncertain that our general method, *without a customized lens*, works better for the task. We thus conducted this study with one of the path tracing tasks designed by Xu et al. [56], which asked the participants to find the length of the shortest path between two randomly-chosen nodes. In a pilot study, we found that the task difficulty increases exponentially with the length of the shortest path. To control the difficulty, we generated random node pairs whose shortest path lengths range from three to five.

**Dataset.** Graph size and cluster structures are two main factors that influence user performance during the path exploration [10,56]. Cluster structures can be measured by the modularity measure [38]. A large modularity means that the graph has clear cluster structures. Du et al. [10] found that the user performance for the three methods P&Z, HF and iSphere shows no significant difference for small graphs. However, iSphere performs better than the others for graphs with low and high modularity, while HF performs significantly worse than the others for exploring large graphs. Based on these findings, we generated a high

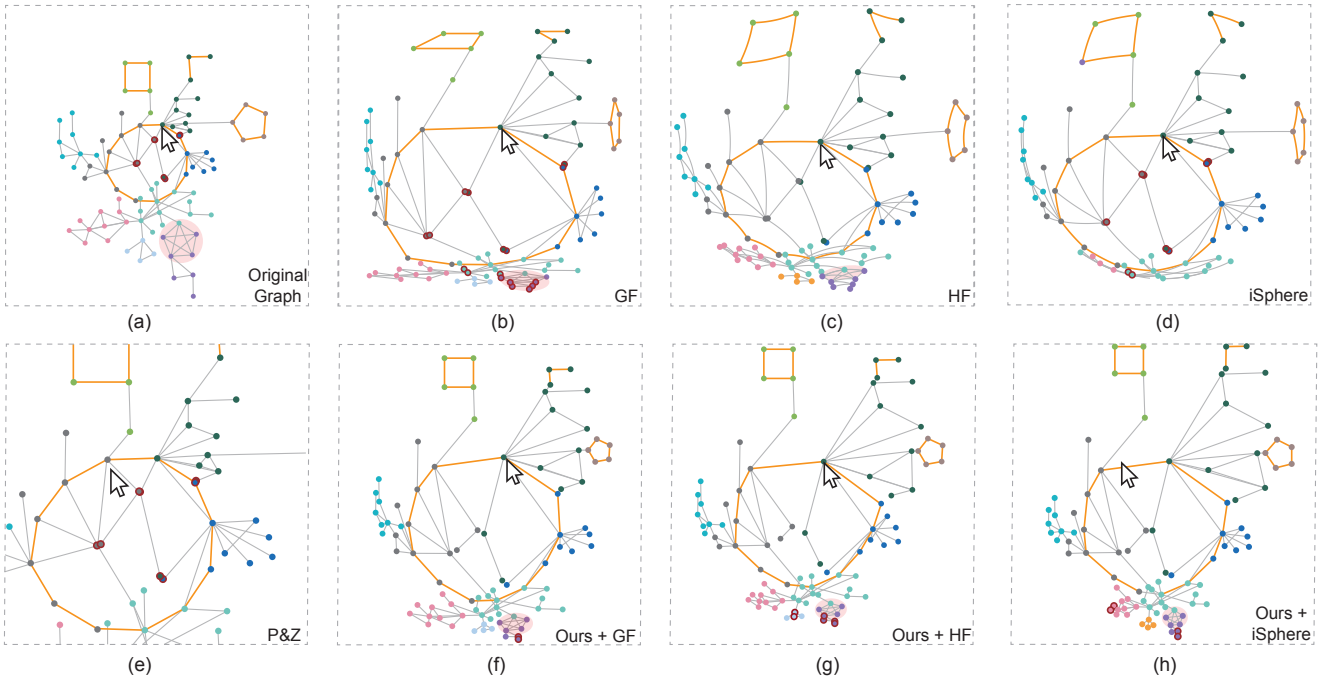


Fig. 8. Visual comparison of various fisheye methods: (a) the original graph; (b) graphical fisheye (GF); (c) hyperbolic fisheye (HF); (d) iSphere; (e) linear magnification; (f) our method with graphical fisheye; (g) our method with hyperbolic fisheye; and (h) our method with iSphere.

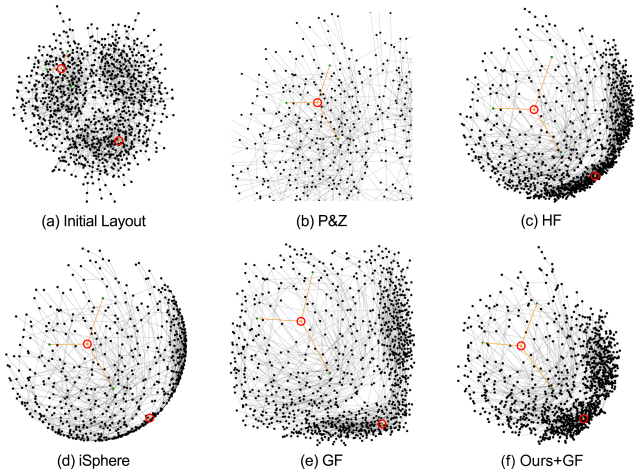


Fig. 9. (a) The graph employed in the lab study; and (b)-(f) the zoom effects of different methods.

modularity graph following Du et al. [10], where the graph has 1024 nodes with the modularity of 0.6. Since the number of possible paths between two nodes grows exponentially with node degrees [7], we chose three as the average node degree to ensure a feasible difficulty level for the participants. Given this graph, we use stress majorization [22] to produce an initial layout, which is taken as the input for all five techniques. In addition, we pre-computed 725 node pairs, among which 90% had shortest edge lengths of five.

**System implementation.** Our system interface in the study supports picking and selection mechanisms to assist the user exploration. In particular, when a user clicked a node with the right mouse button, the node and its adjacent links were highlighted in orange. The highlighting persisted until the user made the next click.

**Participants and apparatus.** We recruited 40 volunteers (24 males and 16 females), aged from 22 to 29 (average 24). The experiment was conducted on a desktop computer equipped with a mouse, a keyboard, and a 24-inch display with 1920×1080 resolution and 144Hz refresh rate. The node-link diagrams were displayed in a window with a white background, where each node was rendered as a black dot and each edge as a black line. Following Du et al.'s advice [10], we used a window size of 150mm×150mm in our experiments. Figures 9(b)-(f) show the results of applying the five techniques to the input graph.

**Procedure.** When the experiment starts, we introduced the five techniques to each participant and demonstrated how they work. Then, the participant was required to perform two practice trials using each of the five techniques. The task was the same as in the actual test: finding the shortest path between a pair of randomly chosen nodes.

In the actual test, each participant was required to perform three trials with each technique, yielding a total number of 15 trials for each participant. We randomized the order of trials for each technique and also the order of the five techniques to avoid bias. Each trial had a 60-second limit, and each participant spent around 15 minutes. Participants could take a break after completing the trials for each technique. After the tasks were completed, we conducted a short interview with the participants, in which we asked them to rate the five methods and explain the reasons behind their ratings.

**Hypotheses.** Based on the abilities of our structure-aware fisheye, we had the following hypotheses before conducting the study:

- ***H1:*** techniques rendered with straight lines (GF, P&Z, and Ours+GF) are more effective than methods rendered with curves (HF, iSphere);
- ***H2:*** fisheye techniques (GF, HF, iSphere, and Ours+GF) are more effective than methods that leave out the context (P&Z); and
- ***H3:*** techniques with a lower distortion or without distortion (Ours+GF and P&Z) are more effective than the methods with strong distortions (GF, HF, and iSphere).

**Analysis.** For each trial, we recorded the task completion time and error, where the error is defined as the absolute difference between the user input path length and the correct path length. Since each participant is required to repeat three trials per technique, we computed the averaged time and error for each technique. In addition, we followed the recommended practices for statistical analysis, and analyzed the results using an estimation-based approach with 95% confidence intervals (CI) [2, 9].

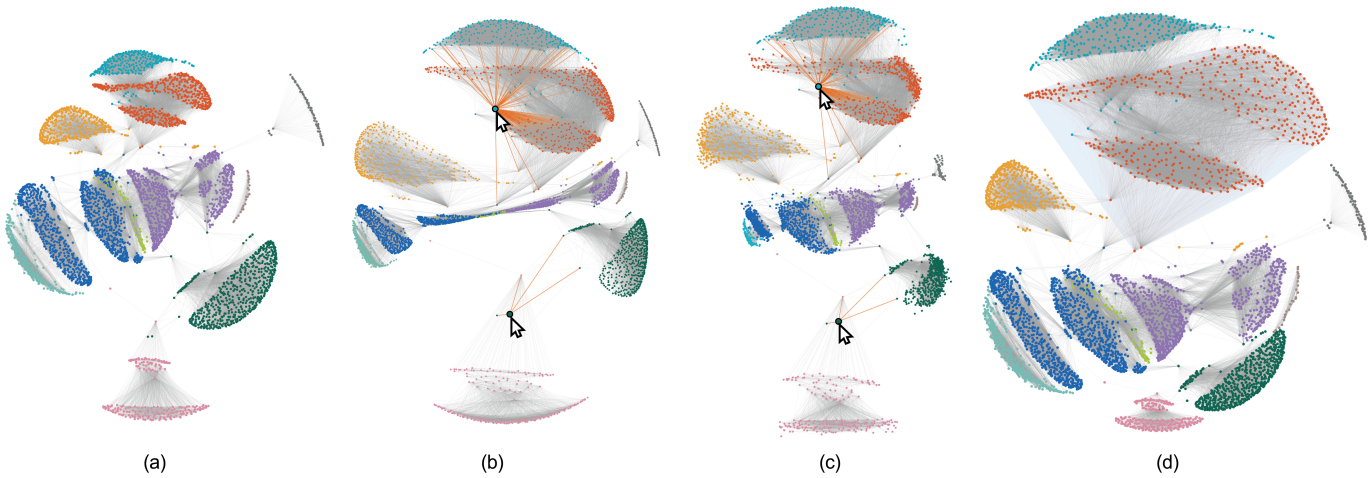


Fig. 11. Exploration on the EGO-NETWORK dataset [35]. (a) Input graph layout. (b) Result generated by the polyfocal projection, where the cluster structures are seriously distorted. (c) Result generated by our polyfocal magnification lens. The two center nodes are highlighted with a red halo, and the edges to their neighbors are shown in yellow. (d) Results generated by our cluster magnification lens. The red focus cluster is highlighted with a gray background, while the shapes of the other clusters are preserved.

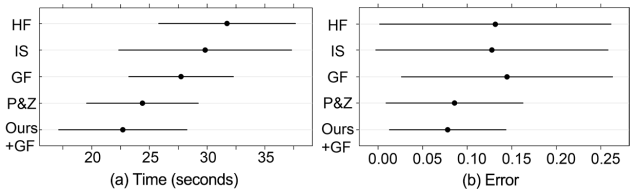


Fig. 10. (a) Mean and 95% CIs of the completion time. (b) Mean and 95% CIs of errors between the true shortest path length and the user input). For both, lower values are better.

#### 6.2.2 Results

The results of our quantitative time and error measures are summarized in Figure 10. Lower values are better for both measured variables. For complete time, Ours+GF is better than all the other methods, especially HF, iSphere (IS) and GF, as shown in Figure 10(a). This supports our hypothesis of H1 and H3, but only partially of H2 because Ours+GF and P&Z have large overlapping confidence intervals. The error measure shows large confidence intervals as shown in Figure 10(b), indicating a less clear result. Nevertheless, the average error is still the lowest for our method – errors were in general small for all methods.

#### 6.2.3 Discussion

**Hyperbolic vs. Pan&Zoom.** The performances of HF and P&Z are as expected and consistent with the user feedback collected by Du et al. [10]. Almost all our users said something along the lines of "*Hyperbolic fisheye results in a cluttered and strongly distorted layout where some edges have large curvatures*," while "*Pan-and-Zoom keeps a stable layout that is intuitive for path tracing, although it loses part of the context.*"

**iSphere.** Du et al. [10] assumed that iSphere performs well for path-tracing related tasks. Our results, however, show that iSphere (IS) performs slightly better than hyperbolic fisheye (HF) but worse than the other three techniques. One user said that "*When zooming into the details of one node in iSphere, the other far-way node is often lost*." This is true because iSphere only renders the nodes shown on the south hemisphere of the displayed ball. To address this issue, the user is often required to do more interactions in order to find and explore the missing nodes. On the other hand, most users are not aware of the distortions produced by iSphere. One user said "*the interaction of iSphere looks like a rotation of a sphere, which is as intuitive as pan-and-zoom.*"

**Graphical fisheye.** Compared to HF and iSphere, GF still uses the straight edges, thus it took longer time to complete the tasks. However, its deformation completely destroys the global structure, so its error is the largest. One user said that "*the interaction with the graphical fisheye is not as intuitive as iSphere and hyperbolic fisheye and the deformations of the graphical fisheye are the most unfamiliar ones.*"

**Our fisheye.** Although our method is based on the geometric transformation of a graphical fisheye, the user performance was found to be the best. Most users said that "*Our structure-aware fisheye combines the advantages of hyperbolic fisheye and pan-and-zoom in a sense that the straight line is helpful for path tracing while the deformation remains small.*" Some users also mentioned that "*the user interaction of our structure-aware fisheye is faster than for the other methods, since clicking a node allows to magnify the structure of interest without pan-and-zoom .*"

### 6.3 Case Studies

We demonstrate the usefulness of our task lens using two real-world datasets: a Facebook social network and a US city network.

**Exploration of Ego-network.** We employ the EGO-FACEBOOK dataset [35] with 4039 Facebook users (nodes) and 88234 relationships between users (edges). Since the network has been grouped into 16 communities, we generate the initial layout by enforcing the cluster non-overlap constraints into the constrained stress model [54], so that different communities are well separated. However, the edges and nodes are still very dense in some areas, making it hard to compare the nodes that are far away from one another. Although simply applying the polyfocal projection can better reveal the local structures around the focal nodes, the structures of different communities might be seriously distorted as shown in Figure 11(b). Here, our polyfocal magnification lens can help. Figure 11(c) shows an example in which two nodes from the inside of two different clusters were selected. We can see that these two nodes both have a few connections to their own cluster but also closely contact with the red and pink clusters.

In an un-magnified layout such as Figure 11(a), it is also hard to figure out the relationships between the nodes. Consider for instance the red cluster at the top, which is separated by some cyan nodes. While it is impossible to see what is going on in an un-magnified view, using our cluster magnification lens on this cluster reveals some interesting structures. From Figure 11(d), we can see that the outlier cyan nodes not only strongly link to the other nodes of this red cluster, but additionally connect to a lot of red nodes. This clearly reveals the relationship between red and cyan clusters in the node-link diagram.

**Exploration of major cities in the United States.** This dataset [42]

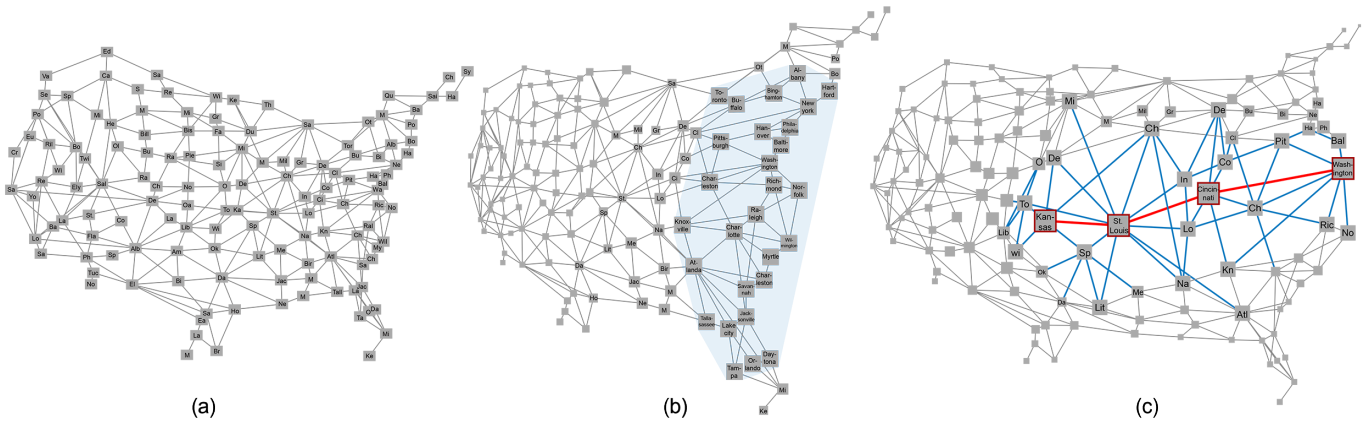


Fig. 12. Exploration of a graph of major cities in the United States [42]. Edges represent the distances or driving time between associated pairs of neighboring cities. (a) Input graph layout. (b) Result generated by our cluster magnification lens, where the focus region is on the light blue background. (c) Result generated by our path magnification lens, where the path from Washington D.C. to Kansas is highlight in red, the child edges of the path in blue, and the edges that are less than $\sigma$ away from the path in blue.

comprises 134 nodes that represent the major cities in the United States and 338 edges that represent the connections between the neighboring cities. The nodes are presented as squares that contain the corresponding city names. The font size of the node labels is adjusted to the distortion factor of the node as suggested by the graphical fisheye [42]. Figure 12(a) shows the input layout. To investigate the dense areas and connections along the east coast, the user applies the cluster magnification lens to this region, resulting in Figure 12(b), where all the node labels can be more clearly shown without overlap.

To find the shortest path from Washington D.C. in the East to Kansas in Midwest, the path magnification lens can be used. To apply the path magnification lens, the user simply selects the two nodes; our interface can automatically generate the layout result shown in Figure 12(c). We can immediately see from the result that there are only two cities between Washington D.C. and Kansas. In addition, it is also much easier to learn the connections around the cities. St. Louis, for instance, has the highest degree among the cities along the path.

**Remark.** In addition, it should be noted that the structure constraints, readability constraints, and temporal coherence constraints in our approach (see Section 4) are incorporated in both case studies. Furthermore, for the effectiveness of the temporal coherency constraints, please watch our supplemental video for the animated results, since we can only present static images in the paper.

## 7 Conclusions and Future Work

We present a structure-aware fisheye technique for graphs. It is based on an optimization objective whose terms correspond to the constraints of structure, readability, and temporal coherence. A number of fisheye lenses are defined for supporting complex graph exploration tasks such as magnifying multiple locations, paths, and whole cluster areas. We evaluated these methods quantitatively by measuring edge orientations and node overlap. In addition, we performed a lab study where the participants explored and searched for shortest paths. Our system outperformed the existing systems in terms of the task completion time and based on the participant interviews, where they preferred the involved interactions above the other methods.

Although most constraints are defined on graph edges, our model is done in the screen coordinate system rather than directly in the graph structure. Doing so, some graph sub-structures related to the focal nodes might not be clearly shown, especially when the related nodes have large Euclidean distances from the focal area. On the one hand, we plan to combine our model with the stress model, so that the graph structures are further enhanced. On the other hand, we would like to extend our method for a wider variety of data sets, such as maps, complex trees and 3D meshes/volumes, by investigating additional constraints.

## Acknowledgments

This work is supported by the grants of the National Key Research & Development Plan of China (2016YFB1001404), NSFC (61772315), NSFC-Guangdong Joint Fund (U1501255), Leading Talents of Guangdong Program (00201509), Shandong Provincial Natural Science Foundation (ZR2016FM12), the Open Research Fund of Beijing Key Laboratory of Big Data Technology for Food Safety, Beijing Technology and Business University, and the Fundamental Research Funds of Shandong University.

## References

[1] J. Abello, S. G. Kobourov, and R. Yusufov. Visualizing large graphs with compound-fisheye views and treemaps. In *International Symposium on Graph Drawing*, pp. 431–441, 2004. doi: 10.1007/978-3-540-31843-9_44

[2] American Psychological Association. *Publication manual of the American psychological association (6th edition)*. American Psychological Association Washington, 2010. doi: 10.24839/1089-4136.jn14.4.133

[3] D. N. Arnold and J. Rogness. Möbius transformations revealed (short film). `https://www.youtube.com/watch?v=lJ5c75k7DP0`, 2008.

[4] A. F. Beardon and C. Pommerenke. The poincaré metric of plane domains. *Journal of the London Mathematical Society*, 2(3):475–483, 1978. doi: 10.1112/jlms/s2-18.3.475

[5] A. Bezerianos, F. Chevalier, P. Dragicevic, N. Elmqvist, and J.-D. Fekete. Graphdice: A system for exploring multivariate social networks. *Computer Graphics Forum*, 29(3):863–872, 2010. doi: 10.1111/j.1467-8659.2009.01687.x

[6] K.-F. Böhringer and F. N. Paulisch. Using constraints to achieve stability in automatic graph layout algorithms. In *Proc. SIGCHI conference on Human Factors in Computing Systems*, pp. 43–51, 1990. doi: 10.1145/97243.97250

[7] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer networks*, 33(1-6):309–320, 2000. doi: 10.1016/S1389-1286(00)00083-9

[8] A. Cohé, B. Liutkus, G. Bailly, J. Eagan, and E. Lecolinet. Schemelens: A content-aware vector-based fisheye technique for navigating large systems diagrams. *IEEE Trans. Vis. & Comp. Graphics*, 22(1):330–338, 2016. doi: 10.1109/tvcg.2015.2467035

[9] G. Cumming. *Understanding the new statistics: Effect sizes, confidence intervals, and meta-analysis*. Routledge, 2013. doi: 10.1111/j.1751-5823.2012.00187_26.x

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

[11] C. Dunne and B. Shneiderman. Improving graph drawing readability by incorporating readability metrics: A software tool for network analysts. *University of Maryland, HCIL Tech Report HCIL-2009-13*, 2009.

[12] 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

[13] T. Dwyer and Y. Koren. Dig-CoLa: directed graph layout through constrained energy minimization. In *Proc. IEEE Information Visualization Symposium*, pp. 65–72, 2005. doi: 10.1109/INFVIS.2005.1532130

[14] 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

[15] T. Dwyer, K. Marriott, F. Schreiber, P. Stuckey, M. Woodward, and M. Wybrow. Exploration of networks using overview+detail with constraint-based cooperative layout. *IEEE Trans. Vis. & Comp. Graphics*, 14(6):1293–1300, 2008. doi: 10.1109/tvcg.2008.130

[16] T. Dwyer, K. Marriott, and M. Wybrow. Topology preserving constrained graph layout. In *International Symposium on Graph Drawing*, pp. 230–241, 2008. doi: 10.1007/978-3-642-00219-9_22

[17] P. Eades, R. F. Cohen, and M. L. Huang. Online animated graph drawing for web navigation. In *International Symposium on Graph Drawing*, pp. 330–335, 1997. doi: 10.1007/3-540-63938-1_77

[18] P. Eades and Q.-W. Feng. Multilevel visualization of clustered graphs. In *International Symposium on Graph Drawing*, pp. 101–112, 1996. doi: 10.1007/3-540-62495-3_41

[19] P. Eades, S.-H. Hong, K. Klein, and A. Nguyen. Shape-based quality metrics for large graph visualization. In *International Symposium on Graph Drawing and Network Visualization*, pp. 502–514, 2015. doi: 10.1007/978-3-319-27261-0_41

[20] G. W. Furnas. Generalized fisheye views. *ACM SIGCHI Bulletin*, 17(4), 1986. doi: 10.1145/22339.22342

[21] G. W. Furnas and B. B. Bederson. Space-scale diagrams: Understanding multiscale interfaces. In *Proc. SIGCHI conference on Human Factors in Computing Systems*, pp. 234–241, 1995. doi: 10.1145/223904.223934

[22] E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *International Symposium on Graph Drawing*, pp. 239–250, 2004. doi: 10.1007/978-3-540-31843-9_25

[23] E. R. Gansner, Y. Koren, and S. C. North. Topological fisheye views for visualizing large graphs. *IEEE Trans. Vis. & Comp. Graphics*, 11(4):457–468, 2005. doi: 10.1109/tvcg.2005.66

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

[25] H. Gibson, J. Faith, and P. Vickers. A survey of two-dimensional graph layout techniques for information visualisation. *Information visualization*, 12(3-4):324–357, 2013. doi: 10.1177/1473871612455749

[26] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Trans. Vis. & Comp. Graphics*, 6(1):24–43, 2000. doi: 10.1109/2945.841119

[27] K. Hornbæk, B. B. Bederson, and C. Plaisant. Navigation patterns and usability of zoomable user interfaces with and without an overview. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 9(4):362–389, 2002. doi: 10.1145/586081.586086

[28] 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

[29] 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

[30] M. Jünger and P. Mutzel. Exact and heuristic algorithms for 2-layer straightline crossing minimization. In *International Symposium on Graph Drawing*, pp. 337–348, 1995. doi: 10.1007/BFb0021817

[31] N. Kadmon and E. Shlomi. A polyfocal projection for statistical surfaces. *The Cartographic Journal*, 15(1):36–41, 1978. doi: 10.1179/caj.1978.15.1.36

[32] 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

[33] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proc. SIGCHI conference on Human Factors in Computing Systems*, pp. 401–408, 1995. doi: 10.1145/223904.223956

[34] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proc. Beyond time and errors: novel evaluation methods for information visualization*, pp. 82–86, 2006. doi: 10.1145/1168149.1168168

[35] J. Leskovec and R. Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Trans. on Intelligent Systems and Technology*, 8(1):1, 2016. doi: 10.1145/2898361

[36] T. Moscovich, F. Chevalier, N. Henry, E. Pietriga, and J.-D. Fekete. Topology-aware navigation in large networks. In *Proc. SIGCHI conference on Human Factors in Computing Systems*, pp. 2319–2328, 2009. doi: 10.1145/1518701.1519056

[37] T. Munzner. Exploring large graphs in 3d hyperbolic space. *IEEE Computer Graphics and Applications*, 18(4):18–23, 1998. doi: 10.1109/38.689657

[38] M. E. Newman. Modularity and community structure in networks. *Proceedings of the National Acad. of Sciences*, 103(23):8577–8582, 2006. doi: 10.1073/pnas.0601602103

[39] K. Perlin and D. Fox. Pad: an alternative approach to the computer interface. In *Proc. SIGGRAPH*, pp. 57–64, 1993. doi: 10.1145/166117.166125

[40] J. Pretorius, H. C. Purchase, and J. T. Stasko. Tasks for multivariate network analysis. In *Multivariate Network Visualization*, pp. 77–95. 2014. doi: 10.1007/978-3-319-06793-3_5

[41] H. Purchase. Which aesthetic has the greatest effect on human understanding? In *International Symposium on Graph Drawing*, pp. 248–261. Springer, 1997. doi: 10.1007/3-540-63938-1_67

[42] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proc. SIGCHI conference on Human Factors in Computing Systems*, pp. 83–91, 1992. doi: 10.1145/142750.142763

[43] M. E. Smoot, K. Ono, J. Ruscheinski, P.-L. Wang, and T. Ideker. Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics*, 27(3):431–432, 2011. doi: 10.1093/bioinformatics/btq675

[44] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. on Systems, Man, and Cybernetics*, 11(2):109–125, 1981. doi: 10.1109/TSMC.1981.4308636

[45] R. Tamassia. *Handbook of graph drawing and visualization*. CRC press, 2013. doi: 10.1201/b15385

[46] P. Ti, Z. Li, Z. Xu, and H. Jia. Optimizing the balance between area and orientation distortions for variable-scale maps. *ISPRS Journal of Photogrammetry and Remote Sensing*, 117:237–242, 2016. doi: 10.1016/j.isprsjprs.2016.03.013

[47] C. Tominski, J. Abello, and H. Schumann. CGV-An interactive graph visualization system. *Computers & Graphics*, 33(6):660–678, 2009. doi: 10.31144/bncc.cs.2542-1972.2014.n37.p163-180

[48] C. Tominski, J. Abello, F. Van Ham, and H. Schumann. Fisheye tree views and lenses for graph visualization. In *International Conference on Information Visualization*, pp. 17–24, 2006. doi: 10.1109/IV.2006.54

[49] Touchgraph Navigator. TouchGraph, LLC, 2009. `www.touchgraph.com`.

[50] F. Van Ham and A. Perer. Search, show context, expand on demand: supporting large graph exploration with degree-of-interest. *IEEE Trans. Vis. & Comp. Graphics*, 15(6), 2009. doi: 10.1109/tvcg.2009.108

[51] J. J. Van Wijk and W. A. Nuij. A model for smooth viewing and navigation of large 2D information spaces. *IEEE Trans. Vis. & Comp. Graphics*, 10(4):447–458, 2004. doi: 10.1109/tvcg.2004.1

[52] F. B. Viegas, M. Wattenberg, F. Van Ham, J. Kriss, and M. McKeon. Manyeyes: a site for visualization at internet scale. *IEEE Trans. Vis. & Comp. Graphics*, 13(6), 2007. doi: 10.1109/tvcg.2007.70577

[53] T. Von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. 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

[54] 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

[55] N. Wong, S. Carpendale, and S. Greenberg. Edgelens: An interactive method for managing edge congestion in graphs. In *Proc. IEEE Information Visualization Symposium*, pp. 51–58, 2003. doi: 10.1109/infvis.2003.1249008

[56] K. Xu, C. Rooney, P. Passmore, D.-H. Ham, and P. H. Nguyen. A user study on curved edges in graph visualization. *IEEE Trans. Vis. & Comp. Graphics*, 18(12):2449–2456, 2012. doi: 10.1109/tvcg.2012.189