TreeNetViz: Revealing Patterns of Networks over Tree Structures

Liang Gou and Xiaolong (Luke) Zhang

Abstract— Network data often contain important attributes from various dimensions such as social affiliations and areas of expertise in a social network. If such attributes exhibit a tree structure, visualizing a compound graph consisting of tree and network structures becomes complicated. How to visually reveal patterns of a network over a tree has not been fully studied. In this paper, we propose a compound graph model, TreeNet, to support visualization and analysis of a network at multiple levels of aggregation over a tree. We also present a visualization design, TreeNetViz, to offer the multiscale and cross-scale exploration and interaction of a TreeNet graph. TreeNetViz uses a Radial, Space-Filling (RSF) visualization to represent the tree structure, a circle layout with novel optimization to show aggregated networks derived from TreeNet, and an edge bundling technique to reduce visual complexity. Our circular layout algorithm reduces both total edge-crossings and edge length and also considers hierarchical structure constraints and edge weight in a TreeNet graph. These experiments illustrate that the algorithm can reduce visual cluttering in TreeNet graphs. Our case study also shows that TreeNetViz has the potential to support the analysis of a compound graph by revealing multiscale and cross-scale network patterns.

Index Terms—Compound Graph, Network and Tree, TreeNetViz, Visualization, Multiscale and Cross-scale.

1 Introduction

Many application domains make use of a compound graph, consisting of two subgraphs of a network and a tree in which the network nodes are the same leaf nodes in the tree. For example, a Java package contains a set of classes with a hierarchical package structure and a class dependency network indicating important relationships among classes. Another example would be a scientific collaboration network, in which researchers are usually affiliated with hierarchical social organizations. Because a network node is mapped to a leaf node in a tree in a compound graph, network links can also imply connections among tree nodes.

Visualizing relationships among nodes in a compound graph with two structures of network and tree could be important. Actors in a social network exhibits a social duality [1], in which each actor can be treated as an individual interacting with other individuals, and also as part of a social group connecting with other groups. The patterns of connections between two individuals, between an individual and a group, or between two groups can provide new insight into social relationships at different levels of social aggregation. For instance, to understand a scientific co-author network, a scholar's activities can be analyzed between different groups, from individual activities to cross-university efforts to international collaborations. The type of analysis enables us to understand an individual's social activities at different levels [2] and also identify the "boundary spanners" in organizations [3].

However, revealing patterns of a network over a tree structure is a non-trivial task. Existing visualization tools for compound graphs fail to fully support exploration of these patterns. Most visualization tools of compound graphs mainly focus on the representations of both tree and network structures. Few of them support aggregation of networks over a tree, interaction and exploration of the patterns of the aggregated networks at different tree levels with an integrated view of both network and tree structures. They fail to answer those questions concerning connections spanning different levels, such as how child nodes under a specified parent node are related to other tree nodes; in what ways connections in two non-leaf tree nodes may differ; and which nodes link two different node groups.

In this paper, we first define a graph model, TreeNet, to represent a compound graph and support multiscale and cross-scale aggregation of a network over a tree with the graph model. We then present a visualization design, TreeNetViz, to support various exploration and interaction of multiscale and cross-scale network patterns in the TreeNet graph. TreeNetViz uses a Radial, Space-Filling (RSF) visualization to represent the tree, a circle layout with novel optimization we proposed to show the aggregated network, and an edge bundling technique to reduce visual complexity. The circular layout algorithm reduces both total edge-crossings and edge length with considerations of the hierarchical constraints and the edge weights in a TreeNet graph.

The paper is organized as the following. Section 2 reviews related literature. Section 3 introduces our TreeNet graph model to represent a compound graph and how to aggregate a network over a tree in an on-demand fashion. Section 4 presents the design and implementation of TreeNetViz based on the graph model, including a novel algorithm to reduce edge crossings. A case study using TreeNetViz to analyze a co-author network is described in Section 5. The paper concludes with future research directions.

2 Related Work

2.1 Compound Graph Visualization

A conventional approach for visualizing a compound graph with a network and a tree is to overlay two types of links in a single view with various strategies. The key idea is to convey both hierarchical information and network connections in a single representation.

Several approaches directly add network connections layered over Treemaps [4, 5]. Fekete et al. [4] present network connections as curves linking the network nodes in a treemap representation. Similarly, ArcTrees [5] combines an arc diagram with a one-dimension treemap to show network connections. The one-dimension treemap is utilized to present hierarchy information and efficiently make use of space compared with a traditional treemap. These approaches are intuitive and straightforward but they do not consider issues like visual cluttering.

Some work follows the same approach of integrating two graphs of tree and network in one view but reduce visual complexity by approaches such as avoiding edge crossings or node occlusion. TimeRadarTrees [6] uses a radial node-link tree to represent hierarchy information with two sets of circle sectors to show network

• *Liang Gou and Xiaolong (Luke) Zhang are with the College of Information Sciences and Technology at the Pennsylvania State University, E-Mail: {lug129, lzhang}@ist.psu.edu.*

Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

connections. The inner circle shows incoming edges and the outer circles represent outgoing edges, and therefore no actual links are used for network connections to avoid edge crossings. However, the design of using spatial information to convey network connections is not intuitive. Hierarchical Edge Bundling (HEB) [7] bundles network edges to avoid edge cluttering with B-Spline curves which use the tree structure as a skeleton. This approach can be applied to different tree representations such as radial circle, treemap and balloon layouts. In our design, we applied this approach to a Radial Space-Filling tree with extra efforts to reduce edge crossings.

Some graph drawing algorithms also deal with a compound graph to reduce edge crossings [8]. In graph drawing, this approach is called hierarchical layout, in which nodes are nested in rectangles to show hierarchy information, and network edges are polylines connecting the rectangles [8, 9]. These approaches emphasize graph aesthetics and do not scale very well even for a graph of moderate size.

Another approach is to use multiple views to draw different graphs separately and then create links between them. VisLink [10] lays out graphs on multiple 2D planes, and then places them in 3D space with links between them. Fung et.al. [11] visualize a set of overlapping networks in 2.5D representation with each network in an individual plane, then using inter-plane edges to represent links between networks. Giacomo et.al. [12] propose an algorithm to lay out two graphs in two views by one-to-many connections with few link crossings, which focuses on the aesthetic criteria of graph drawing. Semantic Substrates [13] places graphs into regions defined by users based on semantic data and allows users to interactively refine the semantic for grouping. This approach separates two interconnected graphs and introduces new links among them. It increases the visual complexity and requires more cognition cost.

Unlike tradtional node-link diagrams, another option is a matrix view combined with other designs. Network relations are shown within adjacency matrix and a hierarchy is added to the circumference along the matrix [14, 15]. Honeycomb [15] also provides multilevel aggregation of network over tree structue. The matrix view eliminates occlusion problems and is good for dense graphs, but it is less intuitive than node-link diagrams [16] and needs extra space to show hierarchical information compared with HEB [7] and our approach.

In summary, the visualization methods of compound graphs discussed above mainly focus on representation of both tree and network structures. Few of them, except Honeycomb [15] with a matrix view, offer the functionality to aggregate a network over a tree upon users' request, and to interact and explore various patterns of aggregated networks at different levels.

2.2 Multiscale Visualization

Multiscale visualization enables people to interact with information space across different scales of analysis, observation, and activity. One underlying technique in multiscale visualization of graphs is hierarchical graph clustering and navigation. The clustering could be topology-based or content-based [17].

Topology-based hierchical graph clustering involves a bottom-up method to draw multilevel clustered graphs [18] and uses a spring-force model to determine locations of node clusters [19]. This approach offers a well-balanced layout of nodes with and between clusters. Some research also addresses interaction and exploration of hierarchical graphs. The DA-TU system [20] supports interactive visualization and navigation of clustered graphs at different hierarchical levels. ASK-GraphView [21] detects hierarchical clusters using a topological feature-based decomposition and allows users to interactively expand child clusters in each cluster. Auber and Jourdan [22] propose a tool that allows users to interactively refine node clusters in a hierarchy. Auber et al. [23] also apply multiscale visualization in displaying small-world networks. Grouse [24] supports interactive exploration of clustered graphs within a hierarchy by expanding or shrinking a node.

Content-based hierarchical graph clustering groups a graph with attribute data about nodes or edges. Some visualization techniques emphasize graph layout based on the attibute data. Wu et al. [25] propose a layout approach to visualize multivariate networks on the surface of a sphere with a Self-Organizing Map. Pretorius and Wijk [26] visualize multivariate state transition graphs by hierarchical clustering based on a user-defined subset of node attributes. The clustering hierarchy is represented by a tree and the aggregated edges are shown as an arc diagram. PivotGraph [27] visualizes graphs by placing nodes on a grid with two specific dimensions from various attributes and aggregating nodes by adding the vaules of two selected attributes. This approach can only deal with two attributes at one time. OntoVis [28] abstracts a social network with an ontology-based schema, then filters and refines the network by desirable elements in the ontology schema. In GrouseFlocks [29], nodes in a hierarchy can be interactively generated and manipulated by user queries. Mizbee [30] is an application to explore conservation relationships in comparative genomics data across a range of scales, from genome to gene.

These approaches and applications are based on the hierarchical structure generated from the network with either topological or content-based clustering. However, the hierarchy information is either hidden in the network structure or not explicitly available to interact and explore [18-20, 22, 25, 28, 30]. Some of them need extra efforts from users to link the hierarchy information with the network structure [24, 29], because two graph structures are not integrated in a single view. None of them provide aggregation views of a network over an existing tree structure in an interactive on-demand style.

3 TreeNet Graph

3.1 TreeNet Data Model

We first give a definition and description of a TreeNet graph model for later discussion. A TreeNet graph is a compound graph consisting of two sub-graphs of tree and network along with a node-mapping schema which defines the relationship between network nodes and tree leaf nodes. A TreeNet graph, *TrN*, is written as:

$$TrN = (T, N, \rho) \tag{1}$$

where $T = (V_T, E_T)$ is a subgraph of tree with a node set, V_T, and an edge set E_T; $N = (V_N, E_N)$ is a subgraph of network with a node set, V_N, and an edge set E_N; ρ is a mapping function: $\rho(n_T) \to n_N$, $n_T \in V_{TL}$, $n_N \in V_N$, and V_{TL} is the set of leaf nodes in the tree T.

A TreeNet graph is a special type of compound graph from previous literature [31]. It should be noted that the mapping function is from a tree leaf node to a network node in a TreeNet graph. It indicates that a tree leaf node can only have one mapped node in the network, but a node in network may have multiple corresponding leaf nodes in the tree.

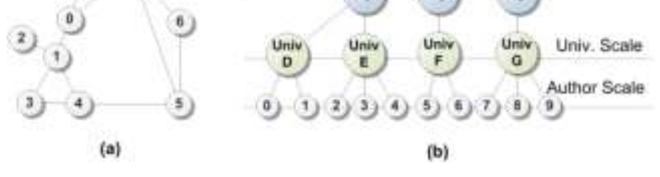

Fig. 1. A TreeNet graph example with (a) a scholar collaboration network, (b) an affiliation tree and node mapping between them.

Fig. 1 shows an example of a TreeNet graph. The TreeNet graph includes a subgraph of a scholar collaboration network in Figure 1a and a subgraph of an afflation tree in Figure 1b. Each node in the collaboration network has affiliation information shown in the tree (the mapping relationship is shown with the same number in the figure). With the above definition, a scholar may have more than one affiliation.

3.2 Multiscale and Cross-scale Network Aggregation over Tree Structure

To analyse a TreeNet compound graph, an important step is to construct different aggregated networks derived from the original network and the tree structure on users' demand.

An aggregated network is generated by applying a cut on a tree to specify which nodes need to be visible, then aggregating the edges based on the nodes in the cut. An aggregation network in TreeNet is constructed as following. We first introduce a cutting line, *CL*, as a set of nodes in the tree, $CL = \{n : n \in V_T\}$. Then, an aggregated network, *AG*, is written as:

$$AG = (V, E) \qquad (2)$$

where nodes, $V \in CL$, and edges, *E*, are aggregated from the network, *N*, with the parent-child relations in the tree ***T***. The aggregated weight of an edge, *e*, as

$$w_{e(u,v)} = \sum\nolimits_{i \in LSet(u),\, j \in LSet(v)} A_{ij} \qquad (3)$$

where $u, v \in V$; A_{ij} is an adjacency matrix of the network *N*, in which a cell a_{ij} is 1 if there is edge between node *i* and *j*; otherwise the element is 0; and *LSet*(*v*) is a function to get the matched network nodes for leaf nodes of a subtree with root node *v*:

$$LSet(v)_{v \in V_T} = \{ t \in V_N : \rho(m), m \in LeafNodes(v) \} . \qquad (4)$$

For a node pair (*u*,*v*) in *AG*, only if the edge weight defined in Equation (3) is larger than zero, we say that an edge exists between the node pair; otherwise, there is no edge between node *u* and *v*. It should be noted that the edge aggregation function shown in Equation (3) is a simple one calculated by the count of underlying edges and can be replaced by other task specific measurements, e.g. betweenness, weight and so on.

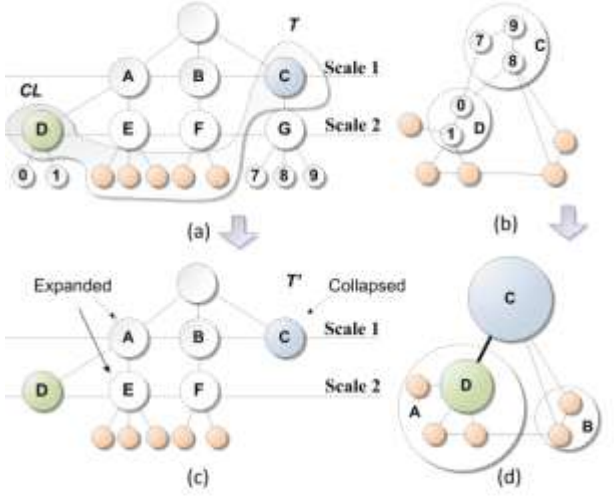

Fig. 2. Network aggregation in a TreeNet Graph.

Figure 2 shows an example of network aggregation in the TreeNet graph of the previous example shown in Figure 1. In Figure 2a, the cutting line (*CL*) includes two non-leaf nodes, corresponding to the two nodes C and D, and five leaf nodes under node E and F. *CL* leads to a different visual structure of the tree (Figure 2c). As a result, the child nodes under nodes C and D are aggregated and invisible, shown as the numbered nodes in Figure 2b. Finally, we get a view of the aggregated network shown in Figure 2d with this cutting. The edge between nodes C and D in Figure 2d is aggregated from the edges (0, 7) and (0, 8) in the original network shown in Figure 2b.

Network aggregation can be ***multiscale*** and ***cross-scale***. A scale is defined by the level (depth) of node in a tree. If nodes in *CL* are from the same level, we say the aggregated network is at this single level. A multiscale network is constructed when we have multiple aggregated networks generated at different single levels. If the *CL* goes through different levels across a tree, the aggregated network is a cross-scale one connecting nodes from various levels. Figure 2d is a cross-scale aggregated network.

4 TreeNetViz: Visualization of TreeNet Graph

A TreeNet graph consists of two sub-graphs of tree and network. To help users understand a compound graph of network and tree, the new visualization design should support:

- *readability tasks in general graphs*, such as identifying graph cardinality (the number of nodes and links), neighbours of a node, following a link and visually searching node and link by labels;
- *network exploration and interaction*, such as helping users to find highly connected nodes, peripheral nodes, connectors between two nodes and closely connected clusters;
- *tree exploration and interaction,* such as conveying parent-child relationship, identifying siblings of a node, common ancestors of nodes and a sub-tree of a node;
- *interactive views of multiscale and cross-scale patterns of the relationships among network nodes over a tree*, such as aggregating or disaggregating a network at the same and different levels over a tree, showing ego networks and critical paths in the multiscale and cross-scale view.

With these requirements, we design and implement TreeNetViz to reveal network patterns over a tree structure. In this section, we present the visualization and interaction design of TreeNetViz, and elaborate a novel algorithm of circular layout to reduce the visual complexity.

4.1 TreeNetViz Visualization

The design of TreeNetViz includes a Radial, Space-Filling (RSF) technique to represent a tree structure, a circular layout to represent an aggregated network, an edge bundling technique to reduce visual complexity and an algorithm to improve circular node placement with the consideration of various constraints. TreeNetViz is implemented with Prefuse Java graph visualization package [32].

4.1.1 Tree as RSF Layout

In TreeNetViz, we use RSF technique to show a tree structure which is also a backbone over which an aggregation network can be placed. The strength of RSF technique is that it can leverage node areas to present additional information about nodes while conveying the parent-child relationship in a tree [33, 34]. Another advantage of a RSF tree in TreeNetViz is that the circular arrangement of nodes in tree is an outline over which an aggregated network can be laid.

The idea of RSF visualization is intuitive: the root node is placed in the centre of a circle; child nodes are assigned within the arc subtended by their parents with angular width which is part of the parent node's width; the angular width angle of a non-leaf node is proportional to aggregation of a property of all its children. In TreeNetViz, the angular width indicates the count of all its children and leaf nodes have a uniform size. The angular width is controllable to show more or less descendant detail, which will be introduced in Section 4.3. Figure 3a shows an example of a RSF visualization of the tree structure in Figure 1b. The node sector color indicates the node scale in the tree structure and the root node is transparent. The hierarchy information of the tree is naturally revealed by this representation. The implementation of RSF is built upon DocuBurst package [35].

4.1.2 Network as Circular Layout

TreeNetViz uses a circular layout to show node connections in a network. A good circular layout can reveal patterns in a graph, such as clusters, ring and star topologies. More important, it naturally uses the circular arcs of generated by a RSF layout and integrates both network and tree structures in a single diagram without introducing duplicated node representations. Our design circularly arranges the

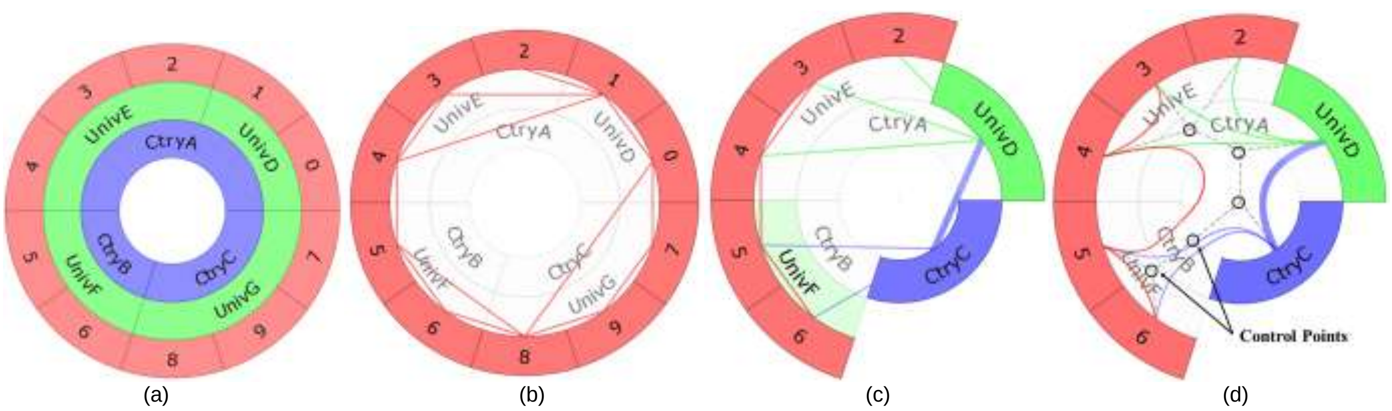

Fig. 3. TreeNetViz Design: (a) a Radial, Space-Filling (RSF) layout of the tree structure; (b) the optimized circular layout of the basic network overlaid on RSF tree; (c) a RSF circular layout of the aggregated network in Fig 2d; (d) the view after edge bundling with β =0.75.

network nodes on the corresponding positions on the circle outlined in RSF and connects node sectors within the circle. The circular layout is also improved with an algorithm discussed in Section 4.4.

Figure 3b and 3c show two examples of circular layout of networks in TreeNetViz design. Figure 3b shows the layout of the original network in Figure 1a with RSF and Figure 3c shows the aggregated network of Figure 2d (note that the circular layouts in Figure 3b and 3c have been optimized with our algorithm introduced in Section 4.4). The expanded parent nodes are transparent and labelled with grey color. The edge uses the same color of the node with the higher scale in the two edge nodes. For example, the edge between node 4 and UnivD has the same color with UnivD which has a higher scale than node 4 in Figure 3c. The line thickness indicates the aggregated weight of an edge. One problem with the straight line of edge in the circular layout is that some edges may be occluded by node sectors. For example, the edge between nodes 2 and UnivD is behind the node sector UnivD in Figure 3c. This issue can be alleviated by the edge bundling technique introduced in the following section.

4.1.3 Edge Bundling

Edge bundling is an effective way to reduce visual cluttering in graph-based visualization [7, 36]. With a special routing approach, edge bundling can also solve the visual occlusion of edge and node sector mentioned above. The edge bundling approach in TreeNetViz is adapted from HEB [7]. The edges are bundled hierarchically with B-Spline curve and the control points are the centres of node sector area in the RSF tree. Figure 3d shows the results of the edge bundling of the network in Figure 3c with bundling strength β =0.75 (a larger β , $\beta \in [0,1]$, yields more curved and closely bundled edges). The control points are circled and highlighted in Figure 3d. We can observe that there is no occlusion among edge and node sector.

4.2 Interactions

In TreeNetViz, we designed several interaction techniques to help explore and analyse a TreeNet graph. The interaction tools include multiscale and cross-scale views with network aggregation, node sector distortion, an ego-network view and a critical path view. The interactions provide functionalities to understand multiscale and cross-scale patterns of a network over a tree in comparison to previous tools [26-30].

Multiscale view. Users can use a slider provided in TreeNetViz to control which scale a network should be aggregated and displayed. This presents aggregated network patterns at different scales of interest. Figure 4a, 4b and 4c show the network patterns at the scales of country, university and individual with the TreeNet graph shown in Figure 1. Note that when the view is drilling down/up along the scale of affiliation, the hierarchical structure of affiliation is also explicitly shown.

Cross-scale view. Users can get a cross-scale view of a network over a tree structure by expanding or collapsing a node. By double

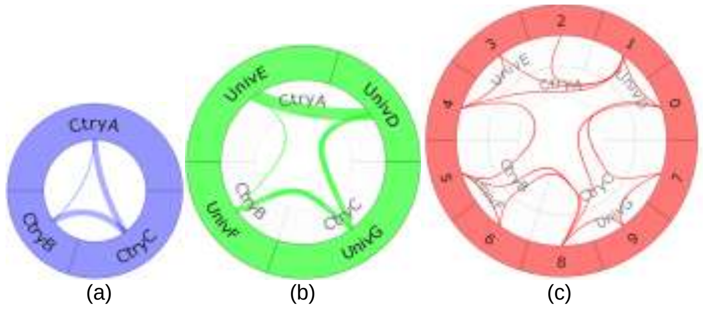

Fig. 4. Aggregated networks at different scales: (a) the country level; (b) the university level; (c) the individual level.

clicking a collapsed non-leaf sector in a RSF tree, users can expand a sector to exam connection patterns of its direct child nodes at a lower scale. This is a drill-down action. Similarly, double clicking an expanded node can collapse its sub-tree and trigger a drill up action. Users can also have multiple nodes expanded or collapsed. Figure 3d is a cross-scale view of the network at three scales. The transparent nodes expand to show details and colored nodes are nodes of interest from the aggregated network.

Node Sector Distortion. Users can dynamically change the angular width of a sector to show different details. Users can increase or decrease the width of a node by scrolling the mouse wheel to up or down when the cursor is hovered over the node. Adjusting a node's angular width affects its sub-tree proportionally and also changes the angular width of its siblings in an oppose manner. This distortion can provide increased details on nodes of interest. More than one node can be adjusted with this interaction. For example, the sector of node CtryA and its children in Figure 5a are enlarged compared with Figure 4c.

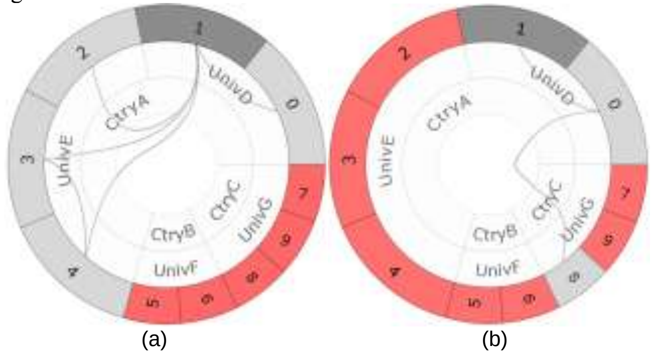

Fig. 5. (a) An ego-network view and (b) A critical path view.

Ego-network View. By right clicking on a node, an ego-network view is activated. An ego-network consists of the direct neighbours of a node of interest and links among them. When the view is activated, the ego-network of the clicked node is highlighted with X-

Ray metaphor (all incident nodes and edges turns into grey and non-incident edges are hidden). Figure 5a shows the ego-network of node 1. This view offers us a clear view of the local network of node 1.

Critical Path View. While an ego-network view shows a local structure of a node, a critical path view illustrates how to reach a target node from a source node. By left clicking a node as the source and left clicking the other node as the target with "Shift" key down, a critical path between them is shown in the view. For example, in Figure 5a, we can see node 1's neighbourhood, but we don't know how to reach node 8 from node 1. Figure 5b shows that node 1 and 8 are connected by node 0.

4.3 Hierarchy-awareness Weighted Circular Layout

In TreeNetViz, one important issue is in what order to place the nodes of an aggregated network along a circle. A good placement of nodes can reduce visual complexity and present patterns of relations in a network saliently. Although the problem of circular layout is studied in previous work [37, 38], the layout presents some new requirements in TreeNetViz:

- *It should consider the restriction of tree structure.* Traditional circular layout methods [37, 38] place the network nodes along a circle at a single level. In TreeNetViz, nodes of an aggregated network, which is generated by expanding or collapsing tree nodes, may be from different levels of a tree. So the circular layout should be restricted by the tree structure.
- *It should consider the weight of edge*. The edge of an aggregated network in TreeNetViz has a weight, which depends on its aggregated value and level on the tree. The edge weight is encoded with line width. Therefore edges with high weight should be addressed properly to avoid visual cluttering in the circular layout. In addition, we may need to avoid edge crossings with the edges from preferable levels, such as lower or higher levels in the tree.

Besides, previous approaches only minimize either the total number of edge crossings [37] or total edge length [38]. However, in some cases, two layouts with same number of edge crossings may have different total edge length (Figure 6 shows such an example). Both the total number of edge crossings and total edge length should be considered in a circular layout.

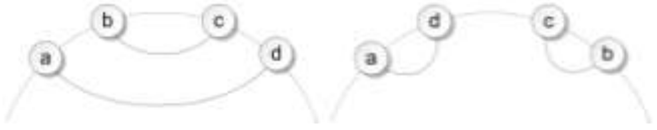

Fig. 6. Two layouts with the same number of edge crossings but different total edge length.

With the observations shown above, we propose an algorithm, Hierarchy-awareness Weighted Circular Layout (HWCL), to place nodes of an aggregated network with the constraints of tree hierarchy and edge weight, and the considerations of both of edge crossings and length.

The basic idea of HWCL is to place network nodes along a circle or an arc to avoid visual cluttering of links among nodes. It first sets criteria of less visual cluttering and then uses a heuristic approach (try different combinations of node order) to achieve a local optimal solution based on the criteria. In HWCL, the criteria of visual cluttering are the combination of the number of edge crossings and edge length.

HWCL first considers tree hierarchy when placing nodes. The child nodes only can be placed and re-ordered under the arc of their parent node. The order of parent nodes must be decided before their children are placed. Only the nodes at the first level (under the root node in the tree) can be placed without the constraint of their parent node. At each step, we place the child nodes only under one parent node. The child nodes under a parent node with larger child count have a higher priority to reorder. With this rule, when users expand or collapse a node, this node will not be re-ordered, its position remains same, and only the child nodes are shifted to reduce the visual cluttering. This can keep the tree structure and make the layout consistent to reduce users cognition cost.

Further, HWCL also utilizes the weight of edge. The idea is that edges with high weight have more visual complexity than those with low weight, because the highly weighted edges have large costs of edge crossings and length. The edge weight is controllable by its aggregated value and level in the tree. The goal is to reduce the total number of crossings and length of highly weighted edges.

4.3.1 Algorithm Background

Suppose we have an aggregated network given in Equation (2), and it is an undirected graph with $n = |V|$ nodes and $m = |E|$ edges. Define a neighbourhood of a node v as $N(v) = \{u \in V : \{v,u\} \in E\}$. We use similar notations in [37]: A configuration of node placement, G, is a position mapping function $\pi(v) = \{i : 0,\ldots,n-1\}$, in which π indicates node positions (either clockwise or counter-clockwise) along a circle. Then, we can define that the order of u is large v in the placement π as (i.e. u is encountered before v in the placement)

$$u \prec_\pi v \Leftrightarrow \pi(v) < \pi(u) \qquad (5)$$

In the placement π, two nodes, u and v, are consecutive, denoted as $u \lhd_\pi v$, if $\pi(v) - \pi(u) = 1$.

4.3.2 Cost Functions

The idea of the algorithm is to place the nodes to minimize the total cost in a placement of nodes. Our assumption is that a good node placement has fewer total edge crossings and short total edge length. Thus, in our approach, the cost function consists of two parts: the total weighted edge crossings and weighted edge length.

We define a weighted crossing as:

$$\chi_\pi(e_1,e_2) = \begin{cases} w(e_1)\cdot w(e_2) & \text{if } u_1 \prec_\pi u_2 \prec_\pi v_1 \prec_\pi v_2 , \\ 0 & \text{otherwise.} \end{cases} \qquad (6)$$

where $e_1 = \{u_1, v_1\}$ and $e_2 = \{u_2, v_2\}$, and $w(\cdot)$ is an edge weight function. The total **cost of weighted edge crossings** in a placement π is:

$$\chi(\pi) = \sum\nolimits_{e_1,e_2 \in E} \chi_\pi(e_1,e_2) \qquad (7)$$

The weighted length of an edge $e \in E$ is defined as

$$\ell_\pi(e) = \begin{cases} w(e)\cdot hop(v,u) & \text{if } hop(v,u) \le hop(u,v) , \\ w(e)\cdot hop(u,v) & \text{if } hop(v,u) > hop(u,v) . \end{cases} \qquad (8)$$

where $hop(v,u) = (\pi(v) - \pi(u))$ mod n.. We use the shorter hops between the two nodes of an edge along the circle as the length metric and weight this length by the edge's weight. Therefore, the total **cost of weighted edge length** in a placement π is:

$$\ell(\pi) = \sum\nolimits_{e \in E} \ell_\pi(e) \qquad (9)$$

The **final cost function** consisting of the two components is written as:

$$\tau(\pi) = (1-\gamma)\cdot\chi(\pi) + \gamma\cdot\ell(\pi) \qquad (10)$$

where $\gamma \in [0,1]$ is a control parameter to balance the weight of edge crossing and length.

We also need control the weight of edges from different levels. In Equation (3) and (5), the weight function is defined as:

$$w(e) = (1-\alpha+\alpha\cdot\log w_e)\cdot(1-\beta+\beta\cdot sqrt(level_e)) \qquad (11)$$

where $level_e$ is the level of the edge e, w_e is the aggregated edge weight, defined in Equation (3), and $\alpha, \beta \in [0,1]$ are parameters controlling the impact of the aggregated edge weight and edge level from the tree structure over the final weight. For example, if the link patterns at lower levels are of interest, we want fewer edge crossings and length with lower level edges, and we can set a large β value.

To sum up, the algorithm goal is to find an "optimal" placement, π_0 , of the graph, G, to minimize the total cost, namely: $\tau(\pi_0) = \min_{\pi} \tau(\pi)$. In this algorithm, we use two-stage optimization to solve this problem.

4.3.3 Two-stage Optimization

To solve the NP-hard problem of circular layout [39], we use a two-stage heuristic optimization derived from paper [37] to minimize the total cost in a placement. The first stage is to initialize the positions of node with certain rules. Then, we follow a greedy strategy to search a locally optimal placement for nodes.

Stage 1: Node Position Initialization

In the first stage, we begin with a single node and append other nodes to the front or end of the placed nodes. Only one node is selected and appended at a time. In the initialization, we need to decide node selection and appending strategies.

Node Selection Strategy. At each step, we choose the node with the largest number of placed neighbours. If two nodes have the same number of placed neighbours, we favour the node with the least number of unplaced neighbours. The rationale of this strategy is to introduce fewer open edges (an *open edge* connects a placed node with an unplaced one), and therefore avoid causing more edge crossings or increasing the total edge length when a new node is placed in the later.

Node Appending Strategy. We append node to the end that results in fewer edge crossings with the open edges. At this stage, we append each node to either the front or the end of the placed node, and do not try every possible position. This is to reduce the computation complexity and further optimization is conducted in the second stage. Note that the crossings with close edges are not considered because they are same for both ends.

Stage 2: Node Sifting Optimization

After nodes are initially placed in a circle, we use sifting to move a node along the circle to find a locally optimal position. The sifting approach was original proposed for binary decision diagrams [40] and used in edge-crossing minimize in circle layout [37].

The idea of node sifting is to iteratively swap a node with its neighbour in one direction, and find the position with the smallest cost shown in Equation 10. Then, we can place the node to the position with the smallest cost. After all nodes have been repositioned, we say ***a round of node sifting*** is completed.

To find the smallest cost for each node, we do not have to calculate the total cost in Equation 10, and only need to calculate the change of cost in each swapping. Because only the positions of the two swapped nodes are changed in each step, the change of the edge crossings and length are only related the two nodes. Thus, we focus on the cost change of the two swapped nodes.

Let the placements before and after swapping as π and π' respectively. The crossing number of two consecutive nodes, u and v, is:

$$c_{uv}(\pi) = \sum_{x \in N(u)} \sum_{y \in N(v)} \chi_{\pi}(\{u,x\},\{v,y\}) \tag{12}$$

The change of crossing cost, Δc , is:

$$\Delta c = c_{vu}(\pi') - c_{uv}(\pi) \tag{13}$$

The total length of a node, u, is:

$$\ell_u(\pi) = \sum_{x \in N(u)} \ell_{\pi}(\{u,x\}) \tag{14}$$

The change of length is

$$\Delta\ell = \ell_u(\pi') - \ell_u(\pi) + \ell_v(\pi') - \ell_v(\pi) \, . \tag{15}$$

The cost change, $\Delta\tau$, in each swapping can get by:

$$\Delta\tau = (1-\gamma) \cdot \Delta c + \gamma \cdot \Delta\ell \, . \tag{16}$$

Thus, in each iteration of node swapping, we can record the cost change, $\Delta\tau$, and then find the position with minimal cost. After each iteration, the node is placed to the locally optimal position obtained above.

In practice, node sifting converges quickly and the computation complexity is acceptable. Usually, a local optimal placement can be achieved by a few rounds of node sifting and each round can be done with $o(nm)$ [37]. The experiments in next section also confirm this.

4.4 Circular Layout Experiments

We conducted several experiments to show the performance of HWCL with variant strategies. The dataset is from a real research field (the background and details of data are introduced in the case study of Section 5.1). The network has 847 nodes and 2,498 edges, and a tree structure with three levels (a root node, 10 nodes at the first level, 90 nodes at the second and 847 at the bottom).

In the following experiments, we started from the aggregated network at the first level (n=10, m=17). Then we expanded the node one by one to the second level. Every time we expanded a new node, TreeNet generated a new aggregated network. We repeated this process at the third level. In this way, we can have hundreds of networks with different numbers of nodes and edges to conduct experiments. Note that the number of nodes in the series of networks is not continuous, because the node number under the expanded node at each step is different.

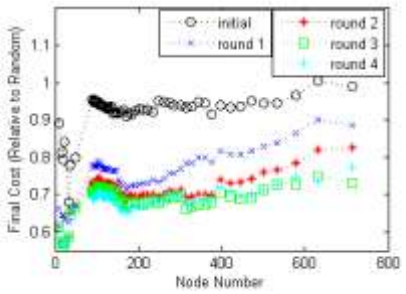

Fig. 7. The final cost of initialization and sifting optimization of different rounds compared to random layout.

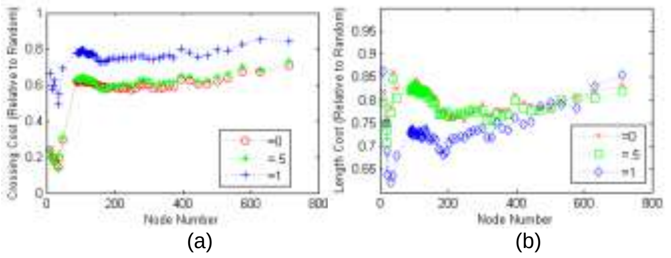

Fig. 8. The costs of edge crossings and total length with γ =0, 0.5 and 1: (a). Relative cost of crossings (b) Relative cost of length.

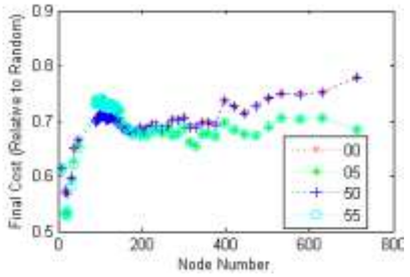

Fig. 9. The final cost with different combinations of edge weight and level parameters.

4.4.1 Algorithm Convergence

The first experiment was to show that the sifting heuristic converges in a few rounds. Figure 7 compares the final cost in Equation (10) after initialization and four rounds of sifting optimization. The final cost is set with $\gamma = 0.5$, the average of the number of edge crossings and the total edge length, and weight parameters are controlled with $\alpha, \beta = 0$. The horizontal axis is the number of nodes (not continuous). The vertical axis is the ratio of the final cost after initialization and sifting optimization to the cost of random layout. As we expected, the final cost is reduced in a few rounds and no obvious improvement after round 4. Thus, the round number is set as 4 in TreeNetViz.

4.4.2 Impact of the Crossing and Length Cost

We also compared the layout results of different combinations of edge crossings and total length used in the final cost with Equation (10). Figure 8a and 8b compare the edge crossing cost and total length cost in the layout optimization with different values of γ =0, 0.5 and 1. The weight parameters are set with $\alpha, \beta = 0$. The vertical axis is the ratio of the cost of our algorithm with different γ to the cost of random layout. We can see that the crossing cost of γ =0 (algorithm using only edge crossings) and γ =0.5 (algorithm using both edge crossings and length) are almost the same in Figure 8a, but in terms of the length cost, γ =0.5 is slightly better than γ =0 as shown in Figure 8b. In addition, Figure 8a shows that only using the length cost (γ =1) can reduce the crossing number (with relative value less than 1), but its effectiveness to reduce crossing is not as good as using edge crossings by comparing γ =1 and 0.5 with γ =0. However, the performance to reduce total length of the algorithm with γ =1 decreases quickly as the number of nodes increases as shown in Figure 8b. We conclude that our algorithm incorporating both edge crossings and edge length can reduce two types of costs than the approach considering only either edge crossings or edge length, but the choice of γ is tricky and needs large scale of experiment, which is beyond the scope of this paper.

4.4.3 Impact of Edge Weight and Level

The last experiment investigated the impact of edge weight and level over layout results. We compared four combinations of level and weight parameters by using unweighted final cost (no weight used in Equation (10), namely $w(e) \equiv 1$) with $\gamma = 0.5$. In Figure 9, the two digital numbers in the legend indicate the combination of weight and level parameters. For example, "00" means $\alpha = 0, \beta = 0$ and "05" for $\alpha = 0, \beta = 0.5$. All combinations generated good cost results compared to the random layout. The two lines with $\beta = 0.5$ have lower cost than the two with $\beta = 0$, which indicates that edge weight can reduce visual cluster. The effect is more obvious as the node number increases. On the contrary, we see that the two lines with different α but same β almost overlap, which shows that impact of level is not obvious.

In summary, the HWCL converges very quickly and the round number of four is used in TreeNetViz. Both the number of edge crossings and total edge length improve the layout results, but there is no general rule how to balance two parts. In our design, we treat them equally with γ =0.5. Finally, the edge weight has larger impact to reduce the final cost compared with edge level and $\alpha = 0.5$ and $\beta = 0.5$ are used TreeNetViz. Figure 10 compares the results of an aggregated network without and with HWCL optimization. It shows that HWCL largely reduces visual cluttering.

5 Case Study: A Co-author Social Network of Medline

In this section, we present a case study using TreeNetViz to analyze a co-author network and help understand collaboration patterns among diabetes researchers at University M. Diabetes research was selected because the topic is studied in many disciplines, ranging

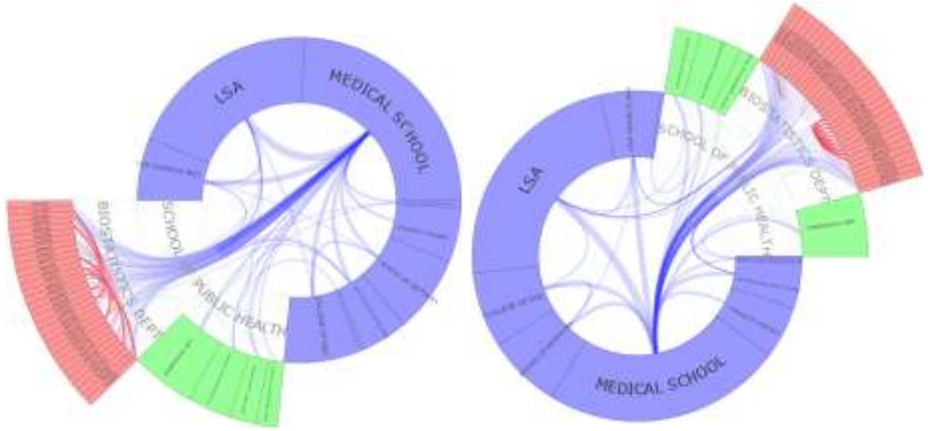

Fig. 10. Comparison of random layout and HWCL with the same aggregated network: (a) Random Layout; (b) HWCL.

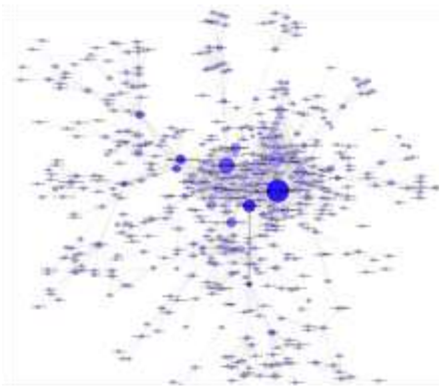

Fig. 11. The largest component in the collaboration network of diabetes researchers.

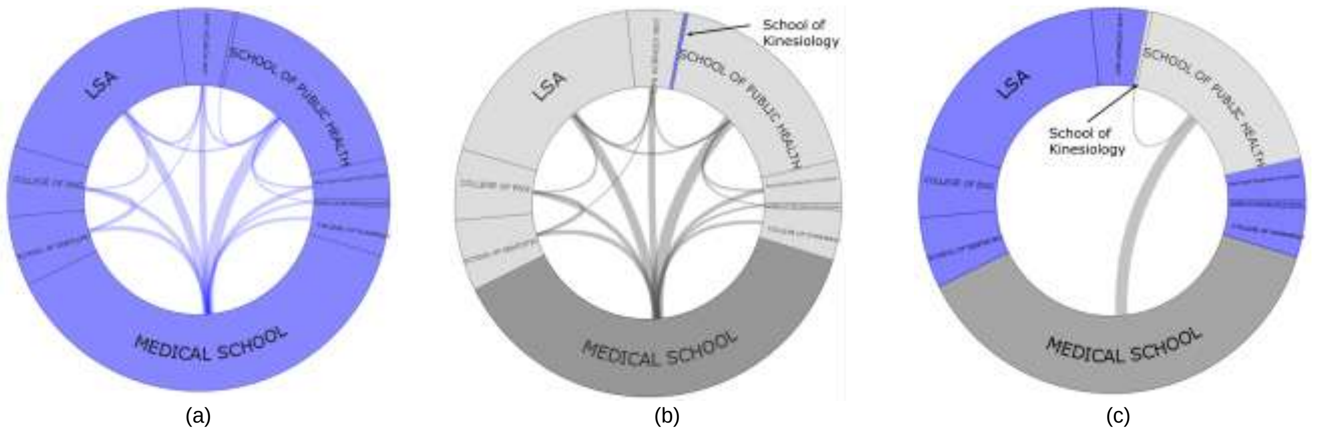

Fig. 12. The collaboration network at the college level: (a) the aggregated network; (b) the ego network of "Medical School"; (c) a critical path between "Medical School" and "School of Kinesiology".

from social sciences to public health to life science and biomedical research. This case study aims at understanding the patterns of peer-to-peer collaboration across organizational boundaries and discovering potential collaborators. While conventional methods provide answers to questions such as who are those most connected authors and how well they are connected (like Figure 11), these methods cannot address questions concerning complex social activities, such as:

- How do collaboration patterns vary across departments and colleges?
- What do cross-department collaboration networks look like? and
- Who are those researchers acting as "boundary liaison" to connect different departments and colleges?

5.1 The Data

Data were collected through two steps. In the first step, primary diabetes terms from MeSH (Medical Subject Headings) [41] were used to search MedLINE research articles published from 2006 to 2010, and a collaboration network was constructed based on the co-author relationship of the retrieved articles. The second step searched the name directory for researchers identified as affiliated with University M and builds a tree structure based on the organizational structure. The names of authors are anonymized. Two collections were combined and cleaned to obtain the final dataset.

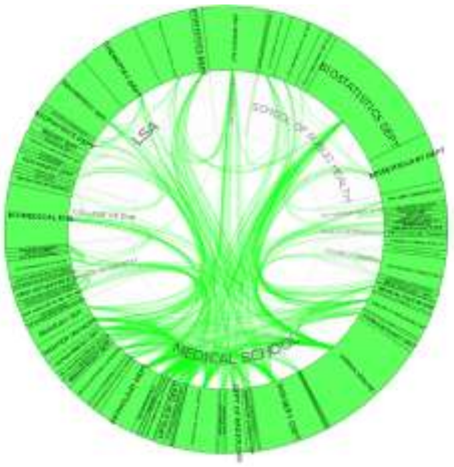

Fig. 13. The view of collaboration among departments.

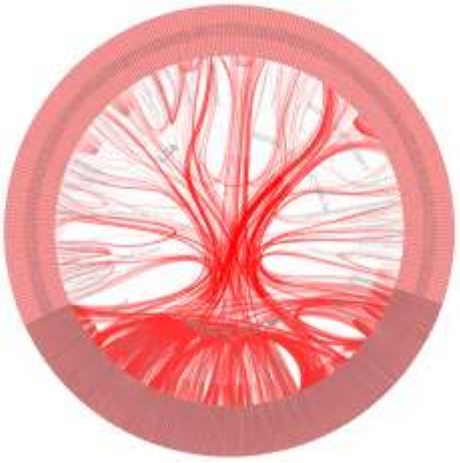
Fig. 14. The view of collaboration among individual researchers.

The dataset includes 614 articles, 847 authors and 2,498 co-author relationships. The largest component of the network is shown in Figure 11. The author affiliation is selected to create a tree with two levels: college and department. TreeNet model identified 10 college-level nodes and 90 department-level nodes. A node at the college level can also be a school or a research centre. Thus, when we use the term "college" henceforth, we also mean school and research centre.

5.2 Multiscale Exploration

With TreeNetVis, users can examine collaboration patterns at three different levels: collaborations involving authors from different colleges, different departments, and also individuals.

The network patterns at different scales can reflect the power and status of collaboration resources, and the access control to social groups and individual authors. For example, in Figure 12a, which shows the collaboration network at the scale of colleges, the size of node sector represents the number of researchers in a group, and thickness of an edge shows the collaboration strength between two groups. From this figure, we can gain some insights into the status of a college in the university collaboration network, such as "Medical School", which not only has the most researchers but also the most active intra-college collaboration activities; "LSA" (Literature, Science and the Arts) and "Public Health" are ranked the second and third, in terms of the number of researchers.

Some structural features on inter-college collaboration are presented in Figure 12b with an ego-network view. For example, Figure 12b is the ego network of "Medical School" whose neighbours are highlighted with X-Ray mode. It shows some cliques at this scale, such as the one consisting of "Medical School", "LSA", "Life Science Institute", and "Public Health". Collaboration between Medical School and Public Health is the strongest. Also, the view helps identify a peripheral player "School of Kinesiology" with blue color. This college is connected to "Medical School" only via "School of Public Health" as shown in Figure 12c.

Changing the level of observation and analysis can provide more details about collaboration at other levels. For example, moving the scale of department, we can see how researchers collaborated across departments (Figure 13). Obviously, the collaboration patterns dominate among several large departments. At the scale of individuals, we can see those active researchers with dense connections, and general collaboration trends are shown in Figure 14.

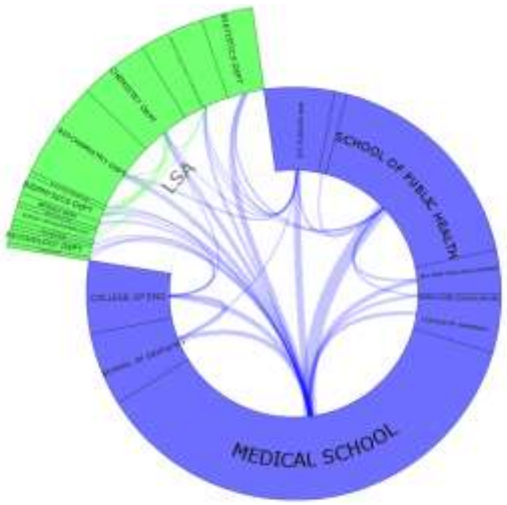

Fig. 15. A cross-scale view of departments under college LSA with with other colleges.

5.3 Cross-Scale Exploration

TreeNetVis allows users to understand co-author patterns across different social levels and identify connectors spanning over different social entities.

Cross-scale views first present patterns how actors collaborate with each other from different scales. Figure 15 shows how the departments in "LSA" cooperated with other colleges. We find that most departments inside this college rarely collaborate with each other, but connect with other outside departments. This view indicates that it might be necessary to further explore why departments do not collaborate and how to motivate local collaborations. Figure 16 shows the collaboration pattern of researchers in "Biochemistry Dept" with all others. These researchers also have more outside connections than internal ones.

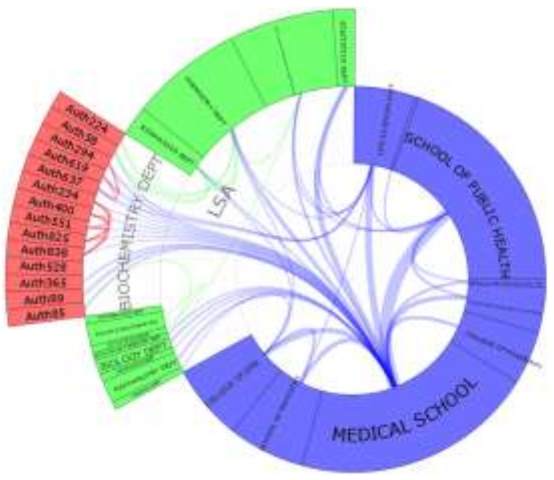

Fig. 16. A cross-scale view with connections over all three levels.

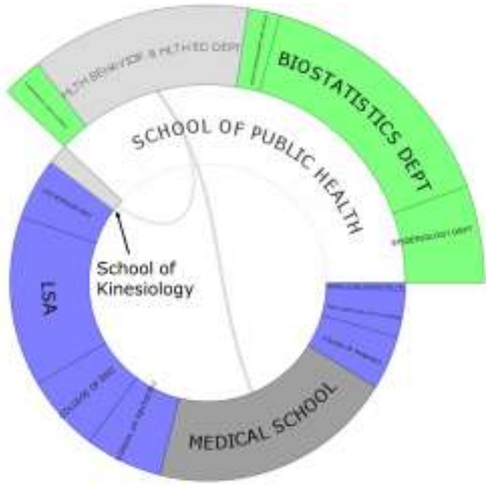

Fig. 17. A liaison at the department level connecting two schools.

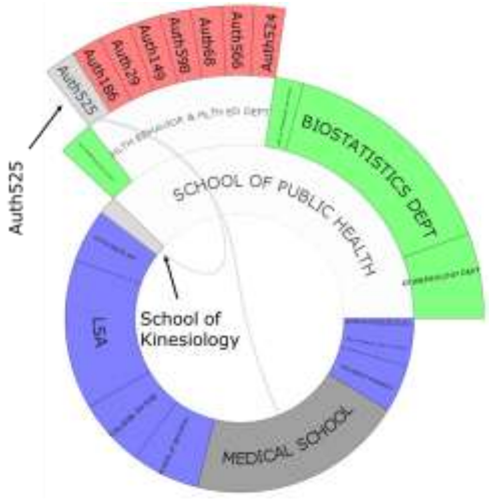

Fig. 18. The researcher "Auth525" on a critical path.

Cross-scale views also help users find out which social actor at one level acts as a "liaison" to link with other actors at another level. Let's go back to the example shown in Figure 12c. We know that "School of Kinesiology" and "Medical School" are connected by "School of Public Health (SPH)". At this point, we may want to know which department in "SPH" connects them. The cross-scale view in Figure 17 shows that "Hlth Behaviour & Hlth ED Dept (HBHED)" at the department level serves as the liaison connecting two schools. Drilling down "HBHED" to individual level (Figure 18), we can find the researcher connecting two different schools is "Auth525".

As shown in this case study, TreeNetVis allows the analysis of complex research collaboration with more depths. With this tool alone, users can examine the relationships among social entities at the levels of college, department and individual, and explore cross-scale connections. TreeNetVis reveals some interesting phenomena that cannot be easily identified with conventional social network analysis tools, such as inactive collaboration among researchers within the same department and key researchers who connect various centres and departments. Such findings lay a foundation for further research to understand questions like:

- what makes an actor become a connector between different organizations;
- what are the barriers to intra- and inter-organizational collaborations respectively; and
- what roles connectors play in collaborative projects ("mailman" or fostering significant sharing and intellectual contribution).

6 Conclusion and Future Work

In this paper, we proposed an approach to model and visualize a compound graph including two subgraphs of tree and network. The TreeNet models the compound graph and supports multiscale and cross-scale network aggregation over the tree structure in the graph. TreeNetViz supports the exploration and interaction of a TreeNet graph by using a Radial, Space-Filling (RSF) visualization to represent the tree structure, a circle layout to show the aggregated network, an edge bundling technique and a novel circular layout algorithm to reduce visual complexity. Our case study of using TreeNetViz to analyze a co-author network indicates the potential of our approach in support of understanding the social network patterns over the affiliation hierarchy.

The research has some limits. First, TreeNetViz still faces the scalability issue as do most graph visualizations. For example, a node sector becomes hard to manipulate when a large number of nodes are arranged along the circle. There is still visual cluttering of edges when lots of edges are incident in a view, although some optimizations of reducing edge crossings and length have been done. Second, current design does not support modifications of the tree structure. In some cases, users may need to merge, add and delete nodes in the tree structure. The modification results in different aggregated networks and connection patterns.

We will extend our work in two directions. First, we will extend the TreeNet graph model and aggregation metrics. A general graph model is desirable to support more diverse compound graphs which do not only consist of two tree and network subgraphs, but also hybrid of them. We will also explore some quantitative metrics to support more complicated analysis tasks in TreeNet graph. For network aggregations, we can provide different approaches for nodes and edges, such as betweenness, eccentricity, authority and hub, and clustering coefficient [42]. Some measures of node similarity are also under development to predict link in the network. Second, we also want to explore other interaction techniques to alleviate the visual complexity. For example, we can provide some other Focus+Context interactions, such as showing the details outside the circle suggested by [33]. For the edge routing, some other strategies to layout edges can also be studied, such as drawing internal edges in a group outside of the circle to avoid crossings.

Acknowledgments

The authors wish to thank Dr. Airong Luo and Ms. Patricia F. Anderson for their suggestions and help on preparation for the data, the anonymous reviewers for their valuable comments.

References

[1] Breiger, R. (1974). The Duality of Persons and Groups. *Social Forces*, 53, pp. 181-190.

[2] Kilduff, M., & Tsai, W. (2003). *Social Networks and Organizations*. London: Sage Publications Inc.

[3] Aldrich, H., & Herker, D. (1977). Boundary Spanning Roles and Organization Structure. *The Academy of Management Review*, 2(2), pp. 217-230.

[4] Fekete, J.-D., Wang, D., Dang, N., Aris, A. & Plaisant, C. (2003). Overlaying Graph Links on Treemaps. In *Proceedings of the 2003 IEEE Symposium on Information Visualization (InfoVis'03)*, Poster Compendium, pp. 82–83.

[5] Neumann, P., Schlechtweg, S. & M. S. T.(2005). ArcTrees: Visualizing Relations in Hierarchical Data. In *Proceedings of the 2005 Eurographics / IEEE VGTC Symposium on Visualization (EuroVis'05)*, pp. 53–60.

[6] Burch, M. & Diehl, S. (2008), TimeRadarTrees: Visualizing Dynamic Compound Digraphs. *Computer Graphics Forum*, 27, pp. 823–830.

[7] Danny, H. (2006). Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, 12, pp. 741-748.

[8] Sugiyama, K. & Misue, K.(1991). Visualization of Structural Information: Automatic Drawing of Compound Digraphs. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(4), pp. 876–892.

[9] Bertault, F. & Miller, M. (1999). An Algorithm for Drawing Compound Graphs. In *Proceedings of the 7th International Symposium on Graph Drawing (GD'99)*, pp. 197–204.

[10] Christopher, C. (2007). VisLink: Revealing Relationships amongst Visualizations. *IEEE Transactions on Visualization and Computer Graphics,* 13, pp. 1192-1199.

[11] Fung, D. C. Y., Hong, S.-H., Koschutzki, D., Schreiber, F. & Xu, K. (2009). Visual Analysis of Overlapping Biological Networks. In *Proceedings of the 2009 13th International Conference Information Visualisation*, IEEE Computer Society, 16(18), pp. 337–342.

[12] Giacomo, E., W. Didimo, Liotta, G. & Palladino, P. (2009). Visual Analysis of One-to-Many Matched Graphs. In *Graph Drawing*, Ioannis G. Tollis and Maurizio Patrignani (Eds.). Lecture Notes in Computer Science, 5417, pp. 133-144.

[13] Shneiderman, B. & Aris, A. (2006). Network Visualization by Semantic Substrates. *IEEE Trans. on Visualization and Computer Graphics*, 12(5), pp. 733-740.

[14] van Ham, F.(2003). Using Multilevel Call Matrices in Large Software Projects. In *Proceedings of the 2003 IEEE Symposium on Information Visualization (InfoVis'03)*, pp. 227–232.

[15] van Ham, F., H.-J. Schulz & Dimicco, M. J. (2009). Honeycomb: Visual Analysis of Large Scale Social Networks. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II (INTERACT '09)*, 5727, pp. 429-442.

[16] Ghoniem, M., Fekete, J.-D., & Castagliola, P. (2004). A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations. In *Proceedings of the 2004 IEEE Symposium on Information Visualization (InfoVis' 04)*, pp. 17-24.

[17] Elmqvist, N., & Fekete, J. (2010). Hierarchical Aggregation for Information Visualization: Overview, Techniques and Design Guidelines. *IEEE Trans. on Visualization and Computer Graphics*, 14(6), pp. 439-454.

[18] Eades, P., & Feng, Q.-W. (1997). Multilevel Visualization of Clustered Graphs. In *Proc. of Symposium on Graph Drawing*.

[19] Huang, M. L., & Eades, P. (1998). A Fully Animated Interactive System for Clustering and Navigating Huge Graphs. In *Proc. of Symposium on Graph Drawing*, pp. 374-383.

[20] Eades, P. (2000). Navigating Clustered Graphs using Force-Directed Methods. *Journal of Graph Algorithms and Applications*, 4(3), 157.

[21] Abello, J., van Ham, F., & Krishnan, N. (2006). ASK-GraphView: A Large Scale Graph Visualization System. *IEEE Trans. on Visualization and Computer Graphics*, 12(5), pp. 669-676.

[22] Auber, D., & Jourdan, F. (2005). Interactive Refinement of Multi-scale Network Clusterings. In *Proc. of InfoVis'05*, pp. 703-709.

[23] Auber, D., Chiricota, Y., Jourdan, F., & Melancon, G. (2003). Multiscale Visualization of Small World Networks. In *Proc. of InfoVis'03*, pp. 75-81.

[24] Archambault, D., Munzner, T., & Auber, D. (2007). Grouse: Feature-Based, Steerable Graph Hierarchy Exploration. In *Proc. of EuroVis'07*, pp. 67-74.

[25] Wu, Y., & Takatsuka, M. (2008). Visualizing Multivariate Networks: A Hybrid Approach. In *Proc. of PacificVIS'08*.

[26] Pretorius, A. J., & Wijk, J. J. V. (2006). Visual Analysis of Multivariate State Transition Graphs. In *IEEE Trans. on Visualization and Computer Graphics*, 12(5), pp. 685-692.

[27] Martin, W. (2006). Visual Exploration of Multivariate Graphs. *Proc. of CHI'06*, pp. 811-819.

[28] Shen, Z., Ma, K. L., & Eliassi-Rad, T. (2006). Visual Analysis of Large Heterogeneous Social Networks by Semantic and Structural Abstraction. *IEEE Trans. on Visualization and Computer Graphics*, 12(6), pp. 1427-1439.

[29] Archambault, D., Munzner, T., & Auber, D. (2008). GrouseFlocks: Steerable Exploration of Graph Hierarchy Space. *IEEE Trans. on Visualization and Computer Graphics*, 14(4), pp. 900-913.

[30] Meyer, M., Munzner, T. & Pfister, H. (2009). MizBee: A Multiscale Synteny Browser. *IEEE Transactions on Visualization and Computer Graphics*, pp. 897-904.

[31] Sugiyama, K. & Misue, K. (1991). Visualization of Structural Information: Automatic Drawing of Compound Digraphs. *IEEE Trans. SMC*, 4(21), pp. 876-893.

[32] Heer, J., Card, S. K., & Landay, J. A. (2005). Prefuse: A Toolkit for Interactive Information Visualization. *Proc. of CHI'05*.

[33] Stasko, J. & Zhang, E. (2000). Focus+Context Display and Navigation Techniques for Enhancing Radial, Space-Filling Hierarchy Visualizations. In *Proceedings of the IEEE Symposium on Information Visualization 2000 (InfoVis '00)*, pp. 57-65.

[34] Yang, J., Ward, M. O. & Rundensteiner. E. A. (2002). InterRing: An Interactive Tool for Visually Navigating and Manipulating Hierarchical Structures. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, pp. 77-85.

[35] Collins, C., Carpendale, S., & Penn, G (2009). DocuBurst: Visualizing Document Content using Language Structure. In *Proceedings of Eurographics/IEEE-VGTC Symposium on Visualization (EuroVis '09)*, 28(3), pp. 1039-1046.

[36] Cui, W., Zhou, H., Qu, H., Wong, P. C. & Li, X. (2008). Geometry-Based Edge Clustering for Graph Visualization. In *Proc. of InfoVis'08*, pp.1277-1284.

[37] Baur, M. & U. Brandes (2005). Crossing Reduction in Circular Layouts. *Graph-Theoretic Concepts in Computer Science*, 3353, pp. 332-343.

[38] Gansner, E. & Y. Koren (2007). Improved Circular Layouts, In *Proceedings of the 14th international conference on Graph drawing (GD'06)*, pp. 386-398.

[39] Masuda, S., Kashiwabara, T., Nakajima, K. & Fujisawa, T. (1987). On the NP Completeness of a Computer Network Layout Problem. In *Proc. IEEE Intl. Symp. Circuits and Systems*, pp. 292–295.

[40] Rudell, R. (1993). Dynamic Variable Ordering for Ordered Binary Decision Diagrams. In *Proc. IEEE Intl. Conf. Computer Aided Design (ICCAD '93)*, pp. 42–47.

[41] Medical Subject Headings. http://www.nlm.nih.gov/mesh

[42] Everett, M., & Borgatti, S. (1999). The Centrality of Groups and Classes. *Journal of Mathematical Sociology*, 23, pp. 181-202.