GraphScape: Integrated Multivariate Network Visualization

Kai Xu* National ICT Australia
Andrew Cunningham† University of South Australia
Seok-Hee Hong‡ National ICT Australia, University of Sydney
Bruce H. Thomas§ University of South Australia

ABSTRACT

In this paper, we introduce a new method, *GraphScape*, to visualize *multivariate networks*, i.e., graphs with multivariate data associated with their nodes. GraphScape adopts a landscape metaphor with network structure displayed on a 2D plane and the surface height in the third dimension represents node attribute. More than one attribute can be visualized simultaneously by using multiple surfaces. In addition, GraphScape can be easily combined with existing methods to further increase the total number of attributes visualized. One of the major goals of GraphScape is to reveal *multivariate graph clustering*, which is based on both network structure and node attributes. This is achieved by a new layout algorithm and an innovative way of constructing attribute surface, which also allows visual clustering at different scales through interaction. A simplified attribute surface model is also proposed to reduce computation requirement when visualizing large networks. GraphScape is applied to networks of three different size (20, 100, and 1500) to demonstrate its effectiveness.

Keywords: Graph visualization, multivariate visualization.

Index Terms: H.5.2 [Information Interfaces And Presentation]: User Interfaces—Theory and methods; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques

1 INTRODUCTION

Many real-world networks are *multivariate*, i.e., they have attributes associated with nodes and/or edges. In this paper, we focus on networks with node attributes only. Examples of such multivariate networks are common in many application domains. For instance, in a social network whose node represents person and edge represents relationship, there is usually information about each person (such as name, age, and gender). Another example from molecular biology is protein-protein interaction network, whose nodes are proteins and edges are reactions. Such networks often have rich information about individual proteins such as name and function.

One of the common tasks in analyzing multivariate networks is *multivariate graph clustering*, i.e., identifying clusters formed by nodes with similar attributes which are not far from each other in terms of graph distance. For instance, in social network analysis, it is interesting to sociologists whether or not people with similar characteristics (node attributes) are also connected to each other. Molecular biologists are interested to know if proteins with similar functions (node attributes) are related, i.e., connected in the protein-protein interaction network.

Graph and multivariate visualization have been well studied individually in the literature. The work by Herman et al. [12] and Wong and Bergeron [21] are excellent surveys on these two topics, respectively. However, there is relatively less work available on multivariate network visualization, which is essential to tasks such as multivariate graph clustering. Two approaches are commonly used in existing methods. The first one is to map attributes to visual elements of a node. A simple example is to map one attribute to node size and another to node color [10]. The second approach, which is an extension of the first one, is to use *glyph* to represent both node and its attributes. One such example is to use a rectangle glyph to replace graph node while the length and width of a rectangle represent two node attributes [4]. Interaction techniques, such as "brushing" [5], can link nodes with their attributes dynamically. However, with such approaches the network structure and its associated data are usually presented in separate views, and some of them only visualize multivariate data on demand. Therefore, we do not consider these approaches in this paper.

There are a few problems associated with existing approaches. For the mapping approach, the visual elements used could interfere each other. For example, when both node size and shape are mapped to attributes, it is difficult to identify node shape when the size is very small. Also, it is difficult to compare the value of attributes represented by different visual elements such as color and size. For the glyph approach, its visual complexity increases quickly as the number of attributes grows. In addition, for large networks, the relatively small size of a glyph makes it difficult to read the information it encodes.

In this paper, we propose a new method called *GraphScape* for multivariate network visualization. GraphScape adopts a landscape metaphor: the network is placed on a 2D plane, and each attribute is represented by a three dimensional surface, whose height indicates its value. GraphScape is designed to assist multivariate graph clustering, which is achieved by its layout algorithm and unique attribute surface construction. These also allow multi-scale visual clustering through interaction. More than one node attribute can be visualized simultaneously, and it can also be easily coupled with existing approaches to further increase the number of attributes visualized. A faster surface model is also proposed to visualize large networks. GraphScape is applied to networks of different scales (20, 100, and 1500) to demonstrate its capability of facilitating multivariate network analysis.

The remainder of the paper is organized as follows. Section 2 provides a brief overview of existing work on multivariate network visualization and methods that employ a landscape metaphor. The details of GraphScape are discussed in Section 3, including layout algorithm and surface construction. The results of applying GraphScape to various networks are presented in Section 4. Section 5 concludes the paper.

2 RELATED WORK

2.1 Multivariate Network Visualization

The HierNet [10] is one of the early adopters of the mapping approach: both the node color and size are used to present attributes. For example, in an email network visualization, the size and color of a node represent the email amount and staff position respectively. The work by Becker et al. [4] is one of the early examples of the glyph approach: the width and height of a rectangular node were used to show the number of incoming and outgoing calls of some major American cities. More recently, both approaches also appeared in "Cytoscape" [16], which is designed for biological net-

*kai.xu@nicta.com
†andrew.cunningham@postgrads.unisa.edu.au
‡shhong@it.usyd.edu.au
§bruce.thomas@unisa.edu.au

Asia-Pacific Symposium on Visualisation 2007
5 - 7 February, Sydney, NSW, Australia
1-4244-0809-1/07/$20.00 © 2007 IEEE

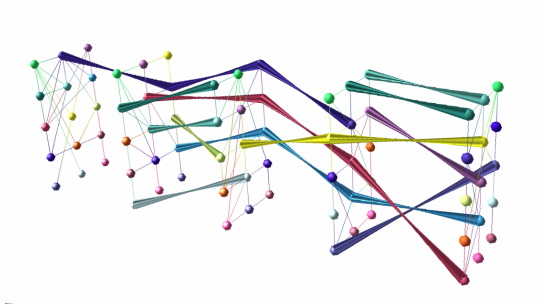


Figure 1: Multiple networks

works. It allows users to customize the mapping between node attributes and visual elements/glyphs.

The work by Saraiya et al. [15] evaluated the effectiveness of existing multivariate network visualization approaches. Tested are the four combinations of one of the two attribute visualizations:

- single attribute (such as node color);
- multiple attributes (glyph).

and one of the two views:

- single view (network structure only);
- multiple views (network structure and linked parallel coordinates).

The evaluation used time-series data so that each node has a series of attribute values, one for each time point. The results suggest that:

- showing one time point (attribute) at a time may lead to more accurate and faster performance for tasks involving analysis of a graph at a single time point, and comparisons between graph nodes for two distinct time points.
- Using glyph may lead to faster performance for tasks involving searching for outlier nodes that show different behavior than the rest of the graph vertices.
- Single views have advantages over multiple views on tasks that require topological information while searching a graph.
- Multiple views are advantageous when analyzing complex behaviors for groups of vertices over time.

New approaches that have recently emerged include the work by Dwyer et al. [8] that used multiple copies of the original network to visualize node attributes, one copy for each attribute. Same node is linked across copies if its attribute values change significantly (Figure 1). The "PivotGraph" [19] provides an aggregated view of a multivariate network. It aggregates nodes and edges that have the same value for the selected attributes, and uses color and size to represent such information (Figure 2).

2.2 Landscape Visualization

GraphScape uses a "landscape" metaphor: a graph is shown in a 2D plane and the height in the third dimension represents attribute value. This metaphor has been previously used to visualize both structural and non-structural data. "ThemeScapes" [20] used this metaphor to provide a visualization of the summary of the text in a digital library (Figure 3(a)). The elevation depicts the number of documents related to a given topic, while other features such as valleys and peaks represent relationships among documents and their sub-topics. However, the collection of documents does not have an

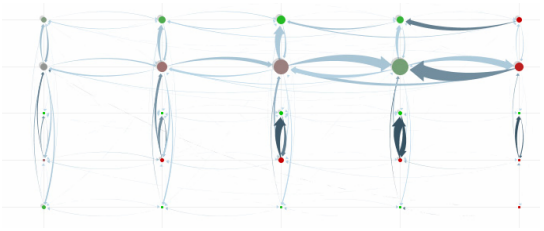


Figure 2: PivotGraph

explicit network structure. "Graph surfaces" [1] adopted a similar metaphor to visualize graph attribute, but the graph is randomly laid out and the topological structure was not shown (Figure 3(b)). Similar approach is also used by Brandes and Willhalm [7] to visualize bibliographic networks (Figure 3(c)). The node height represents its importance in the bibliographic network, which is based on "eigenvector centrality" [6]. However, the network structure is not very clear due to surface occlusion and only one attribute was visualized. Another related work is the "GraphSplatting" [17], in which the node attribute value is mapped to color instead of height (Figure 3(d)). The attribute used is "node density" (number of nodes in a unit display area), which is not a node attribute and affected by layout algorithm. Although the authors tried to visualize more than one attribute by using both color and texture, it is difficult to further extend it.

3 GraphScape

Before the introduction of GraphScape, we provide a formal definition of multivariate network that will be used later in the discussion. A multivariate network is a set

$$G = (N, E, A)$$

where

- $N = \{n_1, n_2, \ldots, n_q\}$ is a set of nodes;
- $E = \{(n_i, n_j) \mid n_i, n_j \in N, i \neq j\}$ is a set of edges;
- $A = \{a_1, a_2, \ldots, a_r\}$ is a set of functions defined on (N, E) so that $a_k : (N, E) \rightarrow \mathbb{R}, 1 \leq k \leq r$. N and E refer to the node and edge set respectively, because an attribute can be a function of properties of other nodes and edges.

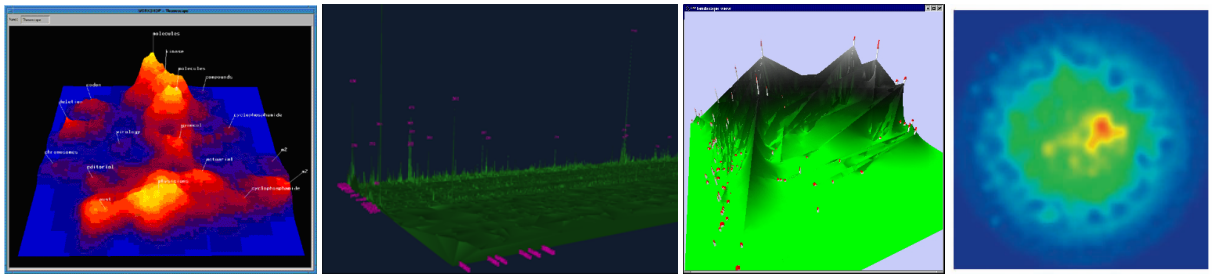
In this paper, we only consider undirected graphs, so (n_i, n_j) is treated as an un-ordered pair. Directed graphs can be visualized with GraphScape by ignoring their edge directions. Also, we only consider continuous attribute function a_k, which includes discrete functions. Categorical values can be mapped to discrete values and then visualized with GraphScape.

A drawing of a graph is a function $\Gamma : N \rightarrow \mathbb{R}^n$ that maps each node n_i to a distinct point $\Gamma(n_i)$ and each edge (n_j, n_k) to a curve with end points $(\Gamma(n_j), \Gamma(n_k))$ in n-dimensional space. For the purpose of GraphScape, we only consider the case of two-dimensional space and straight-line edge. Given the drawing Γ of a multivariate network $G = (N, E, A)$, its GraphScape representation is a set of surfaces

$$S = \{s_1, s_2, \ldots, s_r\}$$

where s_i $(1 \leq i \leq r)$ is a continuous surface in three dimensional space. Each surface s_i represents one attribute function a_i. The position of vertex $v \in s_i$ is a function of graph layout $\Gamma(G)$ and node attribute function a_i, i.e.,

$$\gamma(v) : (\Gamma(G), a_i) \rightarrow \mathbb{R}^3, v \in s_i, 1 \leq i \leq r$$



(a) ThemeScapes (b) Graph surfaces (c) Bibliographic network (d) GraphSplatting

Figure 3: Landscape-metaphor visualization

where $\gamma(v)$ is the position of point v. Sections 3.1 and 3.2 discuss the details of graph layout and surface construction algorithms respectively.

3.1 Layout

The goal of the GraphScape layout algorithm is twofold:

- Achieve conventional graph drawing aesthetics such as reducing edge crossing numbers;
- Show the relationships between node attributes and network structure (multivariate graph clustering).

There are two major challenges to achieve these goals:

- It is computationally very difficult to achieve both goals, given that fulfilling one of the aesthetics is usually NP-complete.
- The two goals can be contradictory in some cases, i.e., a layout that shows the node attributes similarity may not satisfy some of the aesthetic criteria.

Due to these concerns, we decided to base the GraphScape layout algorithm on a force-directed approach, which produces good drawing, is reasonably fast, and can be modified to accommodate new constraints—node attributes similarity in our case.

We chose the spring-electrical force model first proposed by Eades [9]. In this model, edges are springs and nodes are electrically charged particles that repel each other. Precisely, the force on a node n is:

$$\vec{F}(n) = \sum_{(n,n_i)\in E} \vec{f}(n,n_i) + \sum_{n_j\in N, n_j\neq n} \vec{g}(n,n_j) \tag{1}$$

where $f(n,n_i)$ and $g(n,n_j)$ are the spring force and electrical repulsion force respectively. The spring force f increases linearly with the difference between the natural spring length l_0 and the actual distance between n and n_i ($|n-n_i|$), while the electrical force g decreases quadratically as $|n-n_i|$ increases. Precisely,

$$\vec{F}(n) = \sum_{(n,n_i)\in E} k_1 \cdot (|n-n_i| - l_0) \cdot \overrightarrow{n_i n} + \sum_{n_j\in N, n_j\neq n} \frac{k_2^2}{|n-n_j|^2} \cdot \overrightarrow{n_j n} \tag{2}$$

where k_1 and k_2 are two constants that describe the amount of spring and electrical force respectively.

To visually show the node attributes similarity, we decided to place nodes with similar attributes close to each other. This is achieved by introducing two modifications to the previous force model. First, we modified the natural spring length to reflect node attributes similarity. The intuition is that the more similar the two node attributes are, the shorter the length. Precisely,

$$l_0^{n,n_i} = l_{min} + (l_{max} - l_{min}) \cdot \frac{d(n,n_i)}{d_{max}} \tag{3}$$

where l_0^{n,n_i} is the natural spring length between n and n_i. l_{min} and l_{max} are two pre-defined constants, and are the minimal and maximal natural spring length respectively. The main purpose of introducing l_{min} and l_{max} is to avoid cases where two nodes are too close to each other (very small l_{min} value) and very long edges (very large l_{max} value). $d(n,n_i)$ is the *node attributes difference*, which is defined as the following:

$$d(n,n_i) = \sqrt{\sum_{1\le j\le r} (a_j(n) - a_j(n_i))^2} \tag{4}$$

It provides a measurement of node attributes similarity. A large value indicates different node attributes. d_{max} is the maximum node attributes difference of the network. Therefore, the new spring force between a pair of nodes n and n_i is:

$$\begin{aligned} \vec{f}'(n,n_i) &= k_1 \cdot (|n-n_i| - l_0^{n,n_i}) \cdot \overrightarrow{n_i n} \\ &= k_1 \cdot (|n-n_i| - \frac{l_{max}-l_{min}}{d_{max}} \cdot d(n,n_i) - l_{min}) \cdot \overrightarrow{n_i n} \end{aligned} \tag{5}$$

Secondly, we modified the repulsion force between nodes according to their node attributes difference, so that the more similar the two nodes, the less the repulsion force, as follows,

$$\vec{g}'(n,n_i) = (\frac{d(n,n_i)}{d_{max}} + g_0) \cdot \vec{g}(n,n_i) \tag{6}$$

where $\vec{g}(n,n_i)$ is the original repulsion force and g_0 is a small constant to keep nodes with the same attributes apart. Therefore, the new force model is:

$$\begin{aligned} \vec{F}(n) &= \vec{f}'(n,n_i) + \vec{g}'(n,n_i) \\ &= \sum_{(n,n_i)\in E} k_1 \cdot (|n-n_i| - \frac{l_{max}-l_{min}}{d_{max}} \cdot d(n,n_i) - l_{min}) \cdot \overrightarrow{n_i n} \\ &\quad + \sum_{n_j\in N, n_j\neq n} (\frac{d(n,n_i)}{d_{max}} + g_0) \cdot \frac{k_2^2}{|n-n_j|^2} \cdot \overrightarrow{n_j n} \\ &= \sum_{(n,n_i)\in E} k_1 \cdot (|n-n_i| - \frac{l_{max}-l_{min}}{d_{max}} \\ &\quad \cdot \sqrt{\sum_{1\le m\le r} (a_m(n) - a_m(n_i))^2} - l_{min}) \cdot \overrightarrow{n_i n} \\ &\quad + \sum_{n_j\in N, n_j\neq n} (\frac{1}{d_{max}} \cdot \sqrt{\sum_{1\le m\le r} (a_m(n) - a_m(n_i))^2} \\ &\quad + g_0) \cdot \frac{k_2^2}{|n-n_j|^2} \cdot \overrightarrow{n_j n} \end{aligned} \tag{7}$$

The GraphScape layout algorithm based on Equation 7 is outlined in Algorithm 1. The algorithm runs in $O(|N|^2)$ time (where $|N|$ is

the size of N) if the electrical force between every pair of nodes is computed explicitly in step 7. The running time can be reduced to $O(|N|\log|N|)$ if using an improved approach (such as FADE [14]) that does not explicitly compute pairwise node forces. Similarly, it takes $O(|N|^2)$ time to compute the exact value of d_{max}, because it requires comparison of every possible pair of nodes in N. In Algorithm 1, the value of d_{max} is updated during layout computation. This makes it possible to achieve $O(|N|\log|N|)$ time complexity when the improved approach is used. Therefore, the GraphScape layout can be computed in $O(|N|\log|N|)$ time.

Algorithm 1: GraphScape Layout Algorithm

Input: Multivariate network $G=(N,E,A)$
Output: Layout $\Gamma(G)$
1 Assign random position $\Gamma_0(n)$ for every $n \in N$;
2 $d_{max} \leftarrow 0$;
3 **repeat**
4 **for** *every* $n \in N$ **do**
5 Compute the sum of spring force between n and its neighbor n_i ($(n,n_i) \in E$) according to the modified natural spring length l_0^{n,n_i};
6 **if** $d(n,n_i) > d_{max}$ **then** $d_{max} \leftarrow d(n,n_i)$;
7 Compute the sum of electrical repulsion force between n and any other node $n_j \in n$ according to node attributes difference $d(n,n_j)$;
8 **if** $d(n,n_j) > d_{max}$ **then** $d_{max} \leftarrow d(n,n_j)$;
9 Change $\Gamma(n)$ according to its forces;
10 **endfor**
11 **until** $\Gamma(n)$ *converges for every* $n \in N$;
12 **return** $\Gamma(G)$

The introduced modifications of the force model tend to place nodes with similar attributes closer. This addresses the second goal of the GraphScape layout algorithm, while the original spring-electrical force model addresses the first goal. The combined effect is that the new layout algorithm can facilitate revealing clustering in a network based on both topological structure and node attributes, i.e., multivariate graph clustering. This effect is made clear when the node attribute values are visualized with GraphScape surfaces.

3.2 Surface Construction

As previously mentioned, GraphScape consists of a set of three-dimensional surfaces, one for each node attribute. The height of *vertices* (to differentiate from the *nodes* in multivariate network) in a surface is decided by node attributes. Therefore, we name them *attribute surfaces*. A naive way to decide vertex height is to use the attribute value of a node sharing the same x-y location. However, this approach has two possible problems:

- This is essentially the same as approaches that use vertical line glyph to represent attribute value in the third dimension;
- The height of the majority of the vertices, which do not have a corresponding node, remains undecided.

The approach used in attribute surface construction aims to facilitate multivariate graph clustering. This is achieved by allowing the attribute value of a node to affect the height of all the vertices nearby. Therefore, the height of a vertex is the total contribution of the attribute value of surrounding nodes. The height contribution of a node decreases as its distance from the vertex increases. This helps to reveal multivariate graph clustering, because nodes with similar attribute and close in the network tend to form a "visual cluster", which is a part of attribute surface that has similar height but quite different from its surroundings.

Formally, for an attribute $a_i \in A$, the height (z-coordinate) of a vertex v on its attribute surface s_i is:

$$v_z = \mu \cdot \sum_{n \in N} \frac{e(n,v)\cdot a_i(n)}{\max(a_i)} \tag{8}$$

where v_z is the z-coordinate of v, μ is the maximum surface height specified by the user, $\max(a_i)$ is the maximal value of attribute a_i, and function $e(n,v)$ adjusts the contribution of $a_i(n)$ to v_z according to its distance from v. Precisely,

$$e(n,v) = \begin{cases} \frac{1}{2}\cdot(\sin((\frac{|n-v|}{\sigma}+\frac{1}{2})\cdot\pi)+1), & |n-v| \le \sigma \\ 0, & |n-v| > \sigma \end{cases}$$

where $|n-v|$ is the Euclidean distance between n and the projection of v, i.e.,

$$|n-v| = \sqrt{(\Gamma(n)_x - v_x)^2 + (\Gamma(n)_y - v_y)^2} \tag{9}$$

where $\Gamma(n)_x$ and $\Gamma(n)_y$ are the x- and y-coordinate of the layout position of n. The sine function here is to provide a smooth height interpolation (Figure 4). σ is a constant that specifies the radius

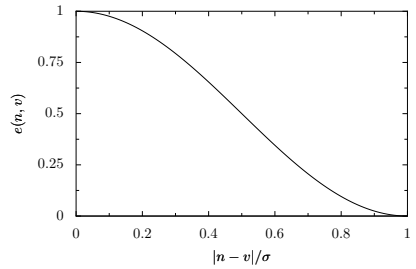


Figure 4: The $e(n,v)$ *function*

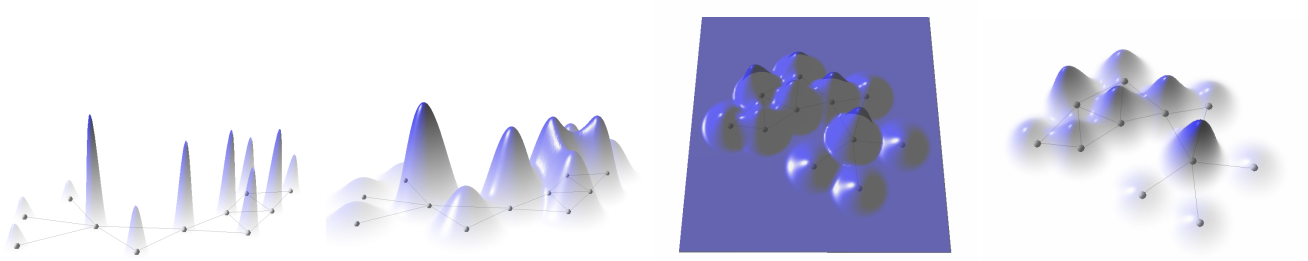
of node attribute contribution. A small σ value means the attribute value of a node contributes to the height of vertices in a small surrounding area, and vice versa. Changing the value of σ has a significant impact on the results of GraphScape. A small radius emphasizes the attribute value of an individual node, which is similar to vertical line glyph; a large radius reveals the overall node attribute distribution, which helps multivariate graph clustering. An example is shown in Figure 5. The same multivariate network is used in both Figure 5(a) and 5(b) where the only difference is the contribution radius. The attribute value of an individual node is clear with small σ value in Figure 5(a), whereas the overall attributes value distribution (high in the middle and right, low on the left) and multivariate graph clustering (nodes on the right have similar attributes value and are also close to each in terms of graph distance) are more obvious with large σ value in Figure 5(b).

During GraphScape design, we found it essential to make attribute surfaces transparent, otherwise it is difficult to perceive the underlying networks. Also, we decided to reduce the transparency as height increases for the following reasons:

- It highlights the part of attribute surface with high value with more opacity;
- The part of attribute surface with zero value is completely transparent, which reduces visual complexity.

Precisely, for any vertex $v \in s_i$, its transparency value α is:

$$\alpha = 1 - \frac{v_x}{\max(v_x)} \tag{10}$$



(a) Small contribution radius (b) Large contribution radius (c) Constant transparency (d) Transparency decreasing with height

Figure 5: Visual parameters of attribute surface

where $\max(v_x)$ is the maximal value of v_x among all the vertices in surface s_i. Here a α value of 1 means completely transparent, whereas a value of 0 means completely opaque. Figure 5 shows a comparison between attribute surfaces with different transparency schemes. It is easy to see that for the attribute surface with constant transparency (Figure 5(c)), it is more difficult to perceive the surface height, visually more complex, and the underlying network is less clear. This makes our proposed model (Figure 5(d)) a better choice.

Each attribute surface is constructed as a 3D grid mesh, and has $O(|N|^2)$ polygons to guarantee a smooth surface. As the network size increases, the larger number of polygons can become expensive to compute, which will result in a slow-down in interaction response time. GraphScape adopted two approaches to address this problem. The first one is to reduce the number of surface polygons by reducing the number of vertices in the attribute surface. An example is shown in Figure 6. The surface resolution can be easily changed through the user interface, or adjusted dynamically by GraphScape. The second approach is to replace the 3D-mesh surface model with a simpler one. We propose a surface model based on Delaunay triangulation: a 2D Delaunay triangulation is performed on the nodes of the multivariate network, then this triangulation is mapped to a 3D polygonal surface whose vertex has the z-coordinate computed by the following formula:

$$v_z = \mu \cdot \frac{a_i(n)}{\max(a_i)} \tag{11}$$

This essentially provides a linear interpolation between neighboring vertices height, which is considerably different from the previous attribute surface model, and so are the resulting visualizations. To make the two surface models as consistent as possible, we add vertices around a node evenly at the distance of contribution radius σ. The number of points added per node is a constant c, and the height of these added vertices are computed according to Equation 8. An example of a triangulated surface is shown in Figure 6(d), whose appearance resembles its smooth-surface counterpart (Figure 6(c)). The surface triangulation is done with Fortune's algorithm [11], which runs in $O(|N|\log|N|)$ time. More importantly, this approach guarantees that the total number of polygons in an attribute surface is $O(|N|)$, because the number of polygons results from Delaunay triangulation is $O(|N|)$ and the number of polygons added at each vertex is a constant.

In theory, GraphScape can visualize arbitrary large number of node attributes by using one surface for each. However, this is difficult to achieve in practice due to problems such as surface occlusion. The problems can be alleviated by using different surface color and/or texture. The details are discussed in the next section along with applying GraphScape to various multivariate networks. Note that it is always possible to combine GraphScape with existing approaches, such as the mapping approach, to further increase the number of attributes it can handle.

4 Application

We applied GraphScape to multivariate networks of three different sizes, namely with 20, 100, and 1500 nodes. These three datasets are social network, synthetic scale-free network, and biological network respectively. We compared GraphScape with the mapping and glyph approach for each dataset except the first one. Due to its small size, we only use the first dataset as an example to demonstrate GraphScape visualization with more than one attribute.

4.1 Small: Social Network

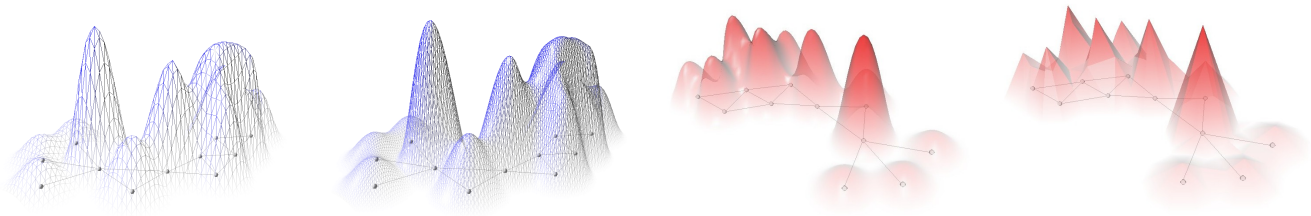
This network is available from the social network analysis book by Wasserman and Faust [18]. The dataset is a collection of graphs commonly known as the "Padgett's Florentine families" networks. The particular one we chose describes the marital relations among 16 most prominent Florentine families in the 15th century. Each family is a node and two families are connected if there is a marriage between them. The two included attributes are "wealth", which is measured in thousands of lira and indicates economic power, and "priorates", which is the number of seats held in the civil council and indicates political power.

The GraphScape visualization of the dataset is shown in Figure 9. The "wealth" is represented as the red surface while the "priorates" is represented as the blue wire-framed surface. An interesting finding that can be drawn from the visualization is that powerful families are most connected in terms of marital relationship. This is shown by the fact that families in the middle—which are more connected—on average have much higher "wealth" and "priorates" values compared to the families in the periphery. This is an example of multivariate graph clustering according to network structure and node attributes. It is interesting to know that there was a big political battle between the "Medicis" and "Strozzis" families, who have the largest "wealth" and "priorates" values respectively, for the control of government. Otherwise, a clearer clustering could be expected.

This is an example of GraphScape with two attributes. During visualization, we found it worked best when both different surface color and texture were applied to differentiate attributes. While it is possible to use only different color or texture, the results were less ideal. It would be interesting to see if GraphScape can handle even more attributes in a similar way, but the lack of alternative surface textures in current GraphScape implementation prevents us from doing so. This will be done once extra textures are available.

4.2 Medium: Synthetic Scale-Free Network

For a medium-sized network, we chose to test GraphScape with *scale-free networks* [3]. The main property of such networks is that



(a) Low resolution (b) High resolution (c) Smooth surface (d) Triangulated surface

Figure 6: Reducing attribute surface polygon number

their node degree distribution follows a "power law", i.e., the number of nodes with degree k is roughly proportional to $k^{-\lambda}$, where λ is a constant. It is found that many real-world networks, ranging from social networks, biological networks, to WWW, are all scale free [3]. The one we used is created by the scale-free network generator available in GEOMI [2]. It follows the "preferential attachment" model [3]: when added to the network, the probability of node n connecting to existing node n' depends on the degree of n'; the higher the degree, the more likely the connection. The generated network has 101 nodes and 153 edges, and we decided to visualize node degree attribute, because it is an important feature for scale-free network.

First, we compared GraphScape with two alternatives. We chose node size for the mapping approach, and vertical line for the glyph approach. The results are shown in Figure 7. Due to the power-law distribution of node degree, there are always a few nodes in the scale-free network that have very high degree. This can be seen clearly from all three visualizations. Another property of scale-free networks is that their diameter is usually small [3]; this is also clearly demonstrated in all three visualizations. Based on these results, we argue that given a medium size network and one attribute, GraphScape is at least as good as the two alternatives.

Secondly, we demonstrate the effects of changing the contribution radius σ. The value of σ is set to small, medium, and large in Figure 8(a), 8(b), and 8(c) respectively. The results demonstrate the "multi-scale clustering" ability of GraphScape: when the σ value is small, many small clusters appear in the network; as σ increases, nodes (small clusters) are grouped into clusters (larger clusters) based on node attributes similarity. This capability provides the possibility of "multi-scale" multivariate graph clustering, which is unique to GraphScape.

Thirdly, we demonstrate combining GraphScape with existing approach and the effect of the new layout algorithm. For the former, we chose to combine with node size mapping, and both represent node degree. The result is shown in Figure 10(a). It is clear that there is interface between the two, and they can be used to complement each other.

For the latter, we show the GraphScape without the new layout, and the result is presented in Figure 10(b). This visualization has the same parameters (such as contribute radius) as the one in Figure 10(a), except its layout is produce by standard spring algorithm. The resulting GraphScape has more clusters, because they are not grouped as what the new algorithm does. In addition, the clusters are closer to each other, because it lacks the repulsion force that changes according to node attributes similarity.

4.3 Large: Biological Network

The third multivariate network we tested with GraphScape is a biological network published in the work by Jeong et al. [13]. This is a protein-protein interaction, i.e., each node is a protein and two proteins are connected if there exists an interaction between them. This a large network with 1458 nodes and 1948 edges. Among the various attributes associated with proteins, we chose to visualize the degree of protein node, because it is found that in yeast there is a high correlation between node degree and lethality, i.e., a protein with high degree (large number of protein-protein interactions) is three times more likely to be lethal (yeast will die if the protein is removed) than those with low degree [13]. Similar to previous tests, the GraphScape is compared against two alternatives: using node size and vertical line to represent node attribute. The results are shown in Figure 11.

It is easy to see that as the network size grows large, it is increasingly difficult to visualize attributes using node size, because their relatively small size makes them difficult to identify and compare (Figure 11(a)). Using vertical line has similar problems. The relative small size of the vertical lines make them difficult to identify. The problem is made worse by the existence of large number of edges, which makes it even harder to differentiate between them and the vertical lines due to overlapping (Figure 11(b)). The GraphScape does not have this problem. It clearly shows the distribution of the attribute value over the network, and highlights the part with large attribute value. It is arguably the most effective of the three methods.

5 Conclusions

In this paper, we propose a new method, GraphScape, for multivariate network visualization. Its layout algorithm and unique surface construction allow it to effectively support multivariate graph clustering. By changing the contribution radius, GraphScape can also visualize clustering at different scales. A faster surface model with $O(|N|)$ polygons is also proposed to reduce interaction response time when visualizing large networks. GraphScape was compared with two alternatives with networks of three different sizes, and it showed a clear advantage over the two other methods when the network size is large.

References

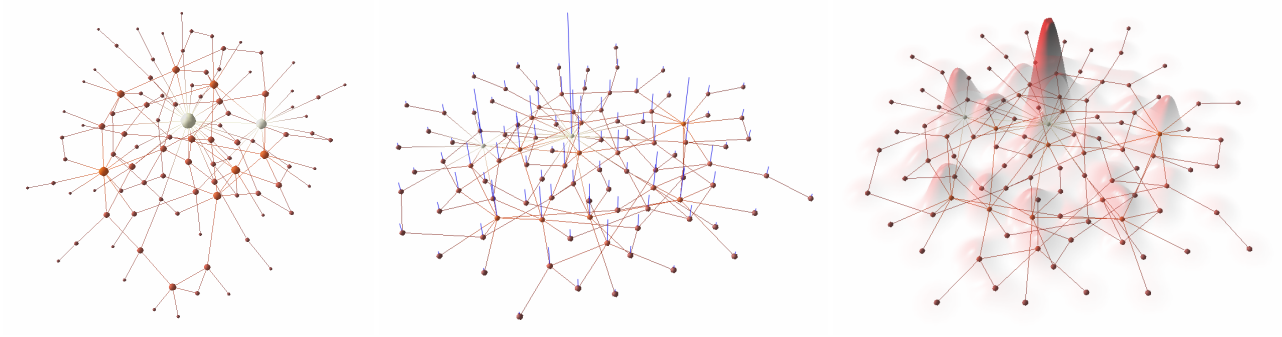
[1] James Abello and Shankar Krishnan. Graph surfaces. In *Proceedings of International Congress on Industrial and Applied Math*, pages 234–244, 1999.

[2] Adel Ahmed, Tim Dwyer, Michael Forster, Xiaoyan Fu, Joshua Ho, Seok-Hee Hong, Dirk Koschützki, Colin Murray, Nikola S. Nikolov, Ronnie Taib, Alexandre Tarassov, and Kai Xu. GEOMI: Geometry for maximum insight. In Patrick Healy and Nikola S. Nikolov, editors, *Proceedings of the 13th International Symposium on Graph Drawing*, volume 3843 of *Lecture Notes in Computer Science*, pages 468–479. Springer, 2005.

[3] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999.

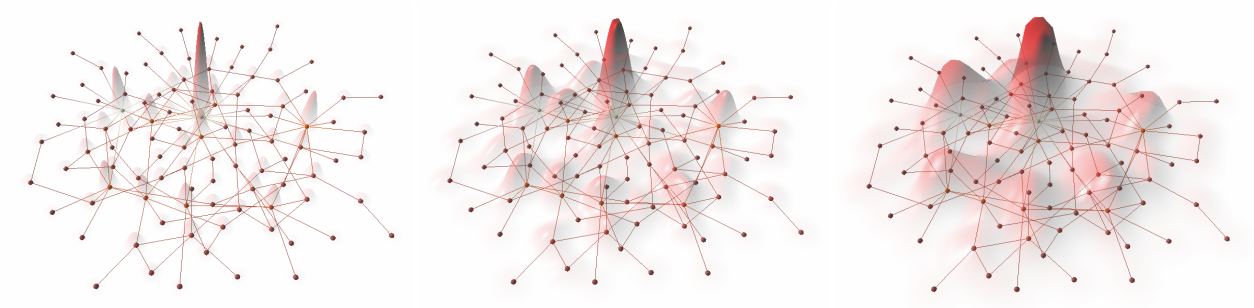
[4] R.A. Becker, S.G. Eick, and A.R. Wilks. Visualizing network data. *IEEE Transactions on Visualization and Computer Graphics*, 1:16–28, 1995.

[5] Richard A. Becker and William S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.



(a) Node size (b) Vertical line (c) GraphScape

Figure 7: Scale-free network with 101 nodes and 153 edges.



(a) Small contribution radius (b) Medium contribution radius (c) Large contribution radius

Figure 8: Various parameter settings of GraphScape.

[6] P Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2:113–120, 1972.

[7] Ulrik Brandes and Thomas Willhalm. Visualization of bibliographic networks with a reshaped landscape metaphor. In *Proceedings of the 4th Joint Eurographics-IEEE TCVG Symposium on Visualization*, pages 159–164, 2002.

[8] Tim Dwyer, Seok-Hee Hong, Dirk Koschuetzki, Falk Schreiber, and Kai Xu. Visual analysis of network centralities. In Kazuo Misue, Kozo Sugiyama, and Jiro Tanaka, editors, *Asia Pacific Symposium on Information Visualisation (APVIS2006)*, volume 60 of *CRPIT*, pages 189–197, Tokyo, Japan, 2006. ACS.

[9] Peter Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[10] Stephen G. Eick and Graham J. Wills. Navigating large networks with hierarchies. In *Proceedings of the 4th conference on Visualization '93*, pages 204–209, 1993.

[11] Steve Fortune. A sweepline algorithm for voronoi diagrams. *ALGORITHMICA*, 2:153–174, 1987.

[12] Ivan Herman, Guy Melancon, and M. Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[13] Hawoong Jeong, Sean P. Mason, Albert-Laszlo Barabasi, and Zoltan N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411:41, 2001.

[14] Aaron J. Quigley and Peter Eades. Fade: Graph drawing, clustering, and visual abstraction. In *8th International Symposium on Graph Drawing*, pages 197–210, 2000.

[15] Purvi Saraiya, Peter Lee, and Chris North. Visualization of graphs with associated timeseries data. In *Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization (INFOVIS'05)*, page 30, Washington, DC, USA, 2005. IEEE Computer Society.

[16] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S. Baliga, Jonathan T. Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13:2498–2504, 2003.

[17] Robert van Liere and Wim C. de Leeuw. Graphsplatting: Visualizing graphs as continuous fields. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):206–212, 2003.

[18] S. Wasserman and K. Faust. *Social network analysis: methods and applications*. Cambridge University Press, 1994.

[19] Martin Wattenberg. Visual exploration of multivariate graphs. In *Proceedings of International Conference for Human-Computer Interaction (CHI)*, page to appear, 2006.

[20] James A. Wise, James J. Thomas, Kelly Pennnock, David Lantrip, Marc Pottier, Anne Schur, and Vern Crow. Visualizing the non-visual: Spatial analysis and interaction with information from text documents. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 51–58, 1995.

[21] Pak Chung Wong and R. Daniel Bergeron. 30 years of multidimensional multivariate visualization. In *Scientific Visualization, Overviews, Methodologies, and Techniques*, pages 3–33, Washington, DC, USA, 1997. IEEE Computer Society.

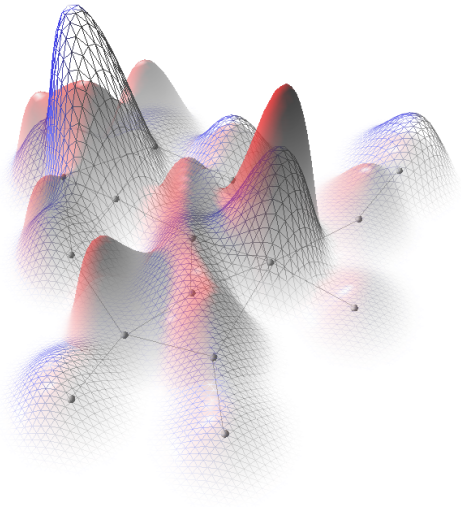
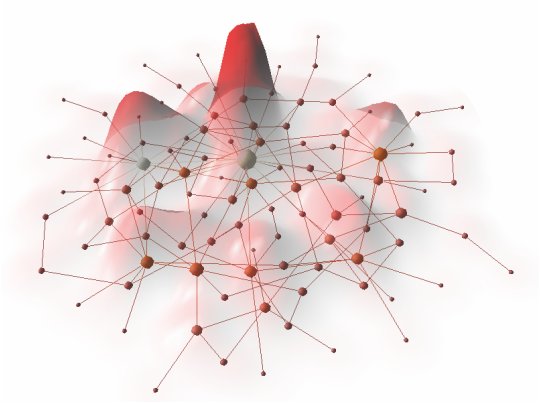
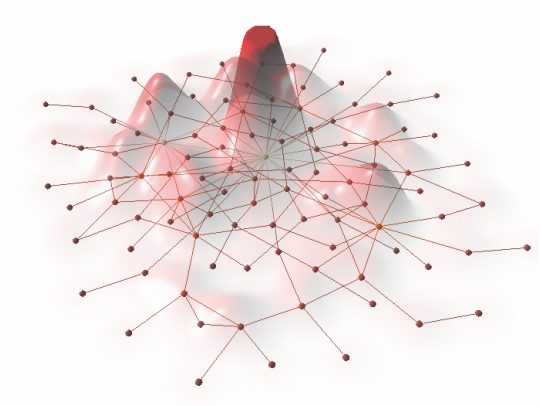


Figure 9: Padgett's Florentine families with "wealth" and "priorates"

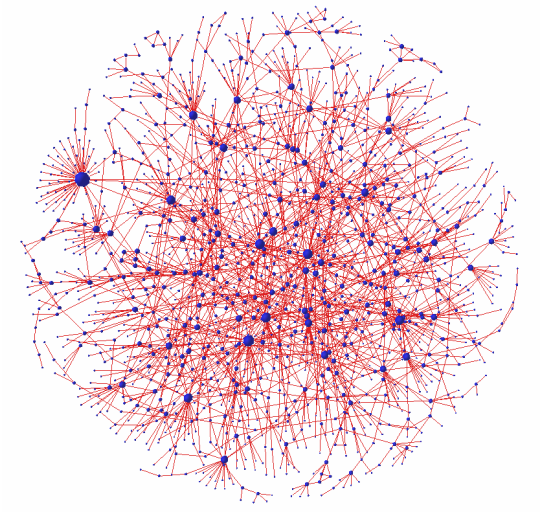


(a) GraphScape with node size mapping

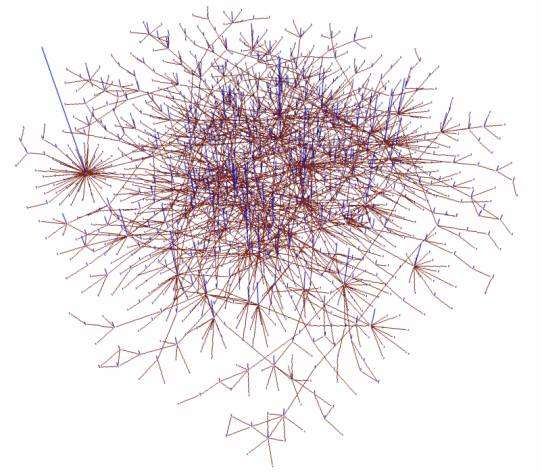


(b) Without GraphScape layout

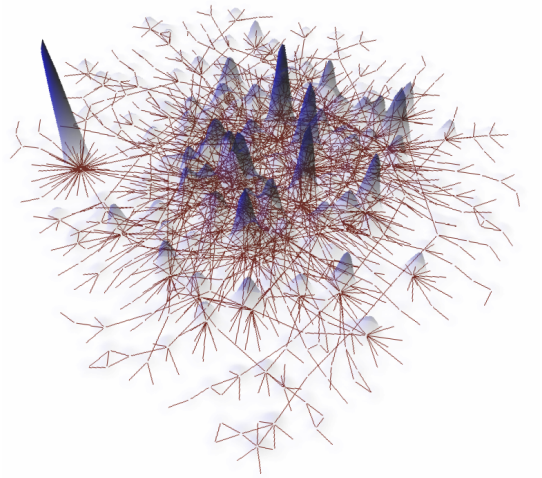
Figure 10: Combination with existing approach and the effect of new layout algorithm



(a) Node size



(b) Vertical line



(c) GraphScape

Figure 11: Protein-protein interaction network with 1458 nodes and 1948 edges.