Drawing Clustered Graphs in Three Dimensions

Joshua Ho[1,2] and Seok-Hee Hong[1,2]

[1] IMAGEN Program, NICTA (National ICT Australia)
[2] School of IT, University of Sydney, NSW, Australia
joshua.ho@student.usyd.edu.au, seokhee.hong@nicta.com.au

Abstract. Clustered graph is a very useful model for drawing large and complex networks. This paper presents a new method for drawing clustered graphs in three dimensions. The method uses a *divide and conquer* approach. More specifically, it draws each cluster in a 2D plane to minimise occlusion and ease navigation. Then a 3D drawing of the whole graph is constructed by combining these 2D drawings.

Our main contribution is to develop three linear time weighted tree drawing algorithms in three dimensions for clustered graph layout. Further, we have implemented a series of six different layouts for clustered graphs by combining three 3D tree layouts and two 2D graph layouts. The experimental results with metabolic pathways show that our method can produce a nice drawing of a clustered graph which clearly shows visual separation of the clusters, as well as highlighting the relationships between the clusters. Sample drawings are available from http://www.cs.usyd.edu.au/∼visual/valacon/gallery/C3D/

1 Introduction

Recent advances in technology have led to many large and complex network models in many domains such as webgraphs, social networks and biological networks. Visualisation can be an effective analysis tool for such networks. Scalability, however, is the most challenging issue, as they may have millions of nodes.

Graph clustering is one of the most efficient approaches to solve the scalability problem. Good clustering methods can identify clusters and the relationships between clusters. Further, many real-world networks have an underlying clustered graph topology.

Recent technological advances in computer graphics hardware made high quality 3D graphics affordable. Further, HCI researchers have established that 3D visualisation can be helpful for giving new insights into abstract data by amplifying human cognition [13].

Clustered graphs have been introduced to both the graph drawing and information visualisation communities. Methods for visualising clustered graphs in two and three dimensions have been developed [6, 7]. Good drawings of clustered graphs should visually separate the clusters effectively as well as reveal the inter-cluster relationships. However, it seems that existing methods fail to satisfy at least one of the following criteria for drawing clustered graphs in 3D:

P. Healy and N.S. Nikolov (Eds.): GD 2005, LNCS 3843, pp. 492–502, 2005.

- minimum number of edge crossings between intra-cluster edges
- minimum volume of the drawing
- minimum sum of inter-cluster edge length
- minimum occlusion views of the drawing
- no overlap between the drawing area of each cluster
- easy navigation

In this paper, we present a new method for drawing clustered graphs in three dimensions. Our work concentrates on layout of flat clustered graphs [7], so we only have to consider one level of clustering. Although this clustered graph model is less general than the common clustered graph model [6], it appears to be a useful model in some real-world applications, like visualization of biological networks. Our proposed method draws each cluster in a 2D plane using an existing 2D drawing algorithm to minimise occlusion and facilitate ease of navigation. Then a 3D drawing of the whole clustered graph is constructed by combining these 2D drawings. For this purpose, we designed three linear time weighted tree drawing algorithms in three dimensions.

Further, we have implemented a series of six different layouts for clustered graphs by combining three 3D tree layouts and two 2D graph layouts. The experimental results with metabolic pathways suggest that it is useful for the visual analysis of large and complex networks.

This separation of dimensionality can help to achieve some of the criteria for clustered graph drawing. In our divide and conquer algorithm, the problem of drawing each cluster in 2D and the problem of arranging each cluster in 3D are addressed separately. However, in order to ensure the overall aesthetics of the drawing of the clustered graph, a post-combination step, called inter-cluster occlusion minimisation, is devised.

Our method also follows Ware's guideline, a $2\frac{1}{2}$ design attitude that uses 3D depth selectively and pays special attention to 2D layout can provide the best match with the limited 3D capabilities of the human visual system" [14].

This paper is organized as follows: the main results of the paper are presented in Section 2. Here we describe a new method for drawing clustered graphs in three dimensions. In particular, three 3D tree drawing algorithms are presented for clustered graph layout. Section 3 presents experimental results and Section 4 concludes.

2 Algorithms for Drawing Clustered Graphs in 3D

First we define our clustered graph model and terminology. We use a flat clustered graph model. That is, we have a set of clusters, $G_1, G_2, \ldots, G_k$ with $G_i = (V_i, E_i)$. Further, we define a supergraph GG such that each G_i is a node v_i in GG, and if there is an edge between a node in G_i and a node in G_j, then there is an edge between v_i and v_j. Note that we can define a weight to each node and edge in GG depending on the size of the cluster and the number of edges between the clusters.

Our algorithm draws a clustered graph using the following four steps:

`Algorithm 3D_Clustered_Graph_Drawing`

1. Draw each cluster in 2D using a 2D drawing algorithm.
2. Draw the weighted supergraph in 3D.
3. Merge the drawings from Step 1 and 2 to construct a 3D drawing of the given clustered graph.
4. Apply inter-cluster occlusion minimisation procedure.

The first step of the algorithm is to draw each cluster in a 2D plane using a 2D graph drawing algorithm. Each node in the cluster is assigned a coordinate in this step. Then the size of each cluster, i.e. the drawing area of the cluster, is computed and assigned as a weight to the corresponding supernode.

In the second step, the layout of the weighted supergraph is computed. As the supernodes have different sizes, a weighted graph layout algorithm is required for this step.

The third step combines the drawings from step 1 and step 2 to construct a 3D drawing of the whole graph. This step is to transform the coordinates of each node to its final position using the coordinates of the corresponding supernode.

The last step of the algorithm is called the inter-cluster occlusion minimisation (ICOM) step. This step addresses the occlusion problem caused by the insertion of inter-cluster edges. We now describe the details of each step.

2.1 Drawing Each Cluster in 2D

The first step is to draw each cluster using any 2D drawing algorithm. One can choose a method depending on the application [5].

Once each cluster is drawn, its size is calculated. More specifically, the size of a cluster is defined by the radius of an enclosing circle of the drawing area.

2.2 Drawing the Supergraph in 3D

The second step is to compute the layout of the supergraph in 3D. In general, any layout algorithm that can draw graphs with different node sizes can be used. In this paper, we mainly concentrate on drawing the supergraph in 3D, where it has a tree structure.

There are two main reasons for focusing on 3D tree drawing methods. First, in general tree layout algorithms are simple and run in linear time. Second, many real life networks resemble tree-like structures, and sometimes visualising the spanning tree of a general graph can be desirable.

In order to accomodate nodes with different sizes, we have designed three linear time 3D tree drawing algorithms, based on the cone tree algorithm [12]: (a) weighted cone tree drawing algorithm (b) weighted rod tree drawing algorithm, and (c) weighted free tree drawing algorithm. We now describe the details of the algorithms.

Weighted Cone Tree Layout. A cone tree [12] is normally computed by a two-pass algorithm: the first pass computes the cone radius by a post-order tree

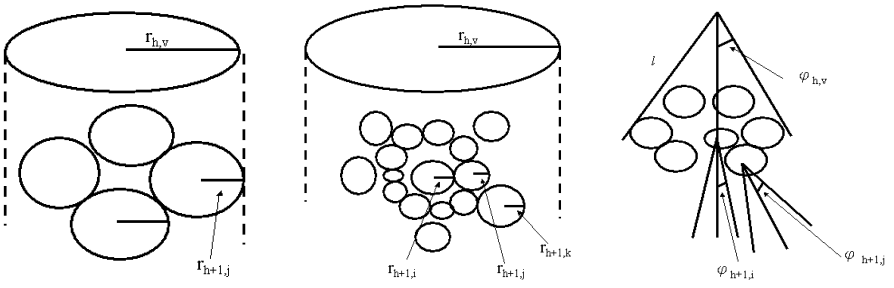

Fig. 1. (a) cone tree (b) rod tree (c) free tree

traversal. In the second pass, a pre-order traversal is used to assign coordinates to each node. For our purpose of drawing weighted trees, we mainly modified the first step.

In a cone tree, each non-leaf node is considered as an apex of a circular cone with cone radius. Let $r_{h,v}$ be the cone radius of the node v at height h. In general, $r_{h,v}$ can be computed from the cone radius $r_{h+1,j}$ of its child nodes j at height $h+1$ using the method described in [4]. However, for our purpose, we need another variable $rad_{h,v}$ to represent the radius (that is weight or size) of the node. In our weighted version of the modified cone tree algorithm, the $r_{h,v}$ of each node is computed as follows (see Figure 1(a)):

$$r_{h,v} = max\{\frac{\sum_j r_{h+1,j}}{\pi} + max_j(r_{h+1,j}), rad_{h,v}\} \tag{1}$$

Weighted Rod Tree Layout. The cone tree algorithm draws all child nodes on the boundary of the circular base of a cone. There is a drawback of this model: a drawing of a highly unbalanced tree may require a large volume. Further, in real world applications, some subtrees may have different levels of importance and it may be desirable to emphasize those subtrees.

In this section, we describe a variation of the cone tree algorithm, called *rod tree*, which chooses one subtree and places it along the z-axis. A similar idea has been explored in the context of visualisation of state transition graphs [9] or symmetric tree drawing [11].

In our rod tree algorithm, we choose a child subtree of maximum height as the center node and then place it on the z-axis. Then, its zero-height siblings form a inner circle to surround the center node. Finally, all the other non-zero height siblings are placed in the outer circle. Therefore, each cone contains at least one child subtree on the axis of the cone, surrounded by at most two concentric circles.

In a rod tree, the child nodes of node v at level h can be divided into three groups: the center node i with the maximum height, a set of nodes j with zero-height, and a set of nodes k with non-zero height (see Figure 1(b)). The cone radius $r_{h,v}$ of node v can be computed as:

$$r_{h,v} = max\{base\_radius(h,v), rad_{h,v}\} \tag{2}$$

The *base_radius* function returns the cone radius. It is computed by first considering the radius of the node i of maximum height, as the inner circle and the circle formed by the zero-height nodes j as outer circle. The radius of the inner circle rad_{inner} is defined as $r_{h+1,i}$. The radius of the outer circle rad_{outer} can be calculated using the normal cone tree method.

Note that the inner and outer circle overlap if $rad_{inner} > rad_{outer} - 2max(r_{h+1,j})$. In this case, we can remove overlapping by increasing the radius of the outer circle. Therefore, rad_{outer} is assigned as $rad_{inner} + 2max(r_{h+1,j})$. The nodes k are then then packed around nodes j. The positions of nodes k are calculated in the same manner as positioning nodes j around node i.

The coordinate assignment step in the rod tree algorithm is similar to the one in the cone tree algorithm.

Weighted Free Tree Layout. Both cone tree and rod tree aim to draw a rooted tree where the hierarchical relationship is important. However, not all clustered graphs have a hierarchical relationship. To cover this case, we now present a 3D free tree drawing algorithm that considers nodes of different sizes.

The main idea is to divide the tree into two subtrees, then draw each subtree in a hemisphere. For each subtree, the layout is computed by placing the child node with maximum height along the axis of the parent node, surrounded by all the other siblings on a spherical surface.

The method for computing the size of a cone is quite different from the one for cone tree and rod tree. In a free tree, each non-leaf node v at level h is an apex of the *spherical* cone. The size of the cone is computed by the angle $\varphi_{h,v}$, an angle between its main axis and the conic surface (see Figure 1(c)).

Let i represent the center node and j represent all other child nodes. Let l be the edge length between adjacent node. The angle $\varphi_{h,v}$ of a leaf node is defined as:

$$\varphi_{h,v} = tan(\frac{2rad_{h,v}}{l}) \tag{3}$$

For all non-leaf nodes, $\varphi_{h,v}$ can be computed as:

$$\varphi_{h,v} = max\{contribution(h+1,i), 2max_j[contribution(h+1,j)]\} \tag{4}$$

where the function contribution() returns a value that contributes to the computation of $\varphi_{h,v}$:

$$contribution(a,b) = max\{\arctan\frac{rad_{a,b}}{l}, \arctan\frac{\tan\varphi_{a,b}}{2}\} \tag{5}$$

Note that the previous steps pack the spherical cones as tightly as possible. In general, a good layout uses the space evenly, therefore a *scaling* procedure is applied. Let $\varphi_{h,available}$ be the total amount of angle available to draw the tree at level h. At each level, $\varphi_{h,v}$ is scaled according to the function:

$$\varphi_{h,v} = \varphi_{h,available} \times \frac{\varphi_{h,v}}{\sum_v \varphi_{h,v}} \tag{6}$$

At the next level $\varphi_{h+1,available} = \varphi_{h,v}$.

2.3 Merging the 2D Drawings into a 3D Drawing

This step combines all drawings of the clusters together according to the supergraph layout. The drawings are transformed to the position specified by the corresponding supernode's position. We used the following two simple methods.

The first method is a simple translation. Since all clusters are initially drawn on the xy plane, all the planes are placed in parallel along the z-axis in 3D.

The second method is a combination of translation and rotation. Each drawing is first translated to the corresponding position, and then rotated towards the center of the drawing. This method places each plane parallel to each other along a concentric sphere.

2.4 The Inter-cluster Occlusion Minimisation Step

The last step of our algorithm considers the placement of inter-cluster edges in order to minimise occlusion and the sum of inter-cluster edge length. A method for Inter-Cluster Occlusion Minimisation (ICOM) is designed for drawing a clustered graph using a 2D spring algorithm for drawing each cluster and one of the 3D tree drawing algorithms. The main idea is to place the nodes connecting an inter-cluster edge close to the boundary of the cluster, as close to the adjacent cluster as possible.

More specifically, our method is based on the spring algorithm with specialized forces. These forces include spring force, repulsion, planar force, ICOM force, and boundary force. The ICOM force can be seen as the spring force for inter-cluster edges. It only pulls nodes that connect to inter-cluster edges along the 2D plane towards the adjacent cluster.

The ICOM force for each node $u \in V$ is defined as follows:

$$f_{icom}(u) = I \sum_{(v \in N)} [\|p_v - p_u\| + refl(\|p_v - p_u\|)] \tag{7}$$

where I is the ICOM force constant and N is the set of nodes connected to u via an inter-cluster edge. The function $refl()$ is a reflection function that returns a vector, a reflection about the cluster plane. Therefore, $\|p_v - p_u\| + refl(\|p_v - p_u\|)$ is a vector parallel to the cluster plane that points to the direction of a neighbour node in the adjacent cluster.

In order to prevent this ICOM force from indefinitely enlarging the cluster, a boundary force is added. The boundary force sets a circular boundary for the cluster. Inside the boundary, a node can move freely. If a node goes over the boundary, a strong force is applied to pull the node back. More specifically, the boundary force for each node $u \in V$ is defined as follows. If $\|p_u\| > bound$, then:

$$f_{boundary}(u) = -Bp_u \tag{8}$$

where p_u is defined as the position of node u. If u is out of the circular boundary of radius bound, a force of $-Bp_u$ is applied to pull the node back inside the boundary. B is a positive boundary force constant.

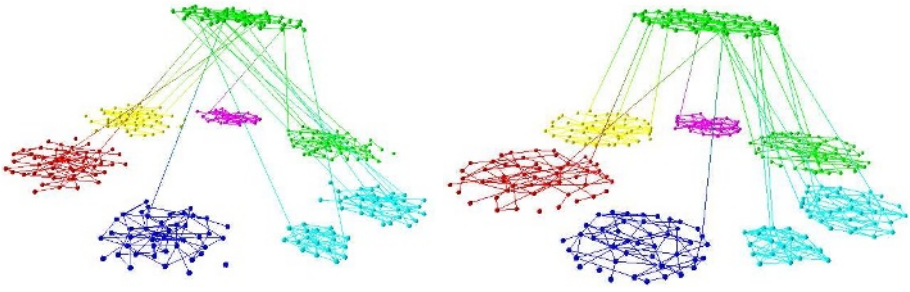

Fig. 2. Comparison of clustered graph layout (a) without ICOM (b) with ICOM

For example, see Figure 2. A randomly generated clustered graph of 329 nodes, 674 edges and 8 clusters is drawn with the weighted cone tree and spring algorithm. ICOM allows a node that is connected to another node in an adjacent cluster to be drawn closer along its own 2D plane. This reduces the visual complexity of the whole drawing by less occlusion and shorter inter-cluster edges. Since the boundary force is applied to restrict the size of the cluster, all resulting clusters are circular in shape. The normal spring, repulsion and planar force allow each cluster to optimise the aesthetic criteria.

2.5 Extension to the General Case

Not all clustered graphs have a supergraph of tree structure. In this case, the maximum spanning tree of the supergraph can be used. Since the weight of a superedge is the actual number of edges between the two clusteres it connect to, the use of a maximum spanning tree ensures most of the inter-cluster edges are drawn between nodes on different tree levels.

3 Implementation and Experimental Results

Our algorithm has been implemented as a set of plug-in modules for GEOMI, a visual analysis tool for large and complex networks [1]. More specifically, we have implemented a series of six different layouts for clustered graphs by combining three 3D tree layouts (weighted cone tree, weighted rod tree and weighted free tree) and two 2D graph layouts (force directed layout and circular layout).

For data sets, we have used both randomly generated data sets and real world data sets. For this purpose, we implement a generator to randomly generate clustered trees, see [1].

Figure 3, 4, 5 shows three different layouts of clustered graphs. From the drawing, it is easy to identify each cluster separately. Further, it is easy to understand the relationship between clusters.

For real world networks, we used metabolic networks. A metabolic network is a collection of chemical reactions in cells. Visualisation of these networks can help identify key features and possible malfunctions of cells. The whole metabolic

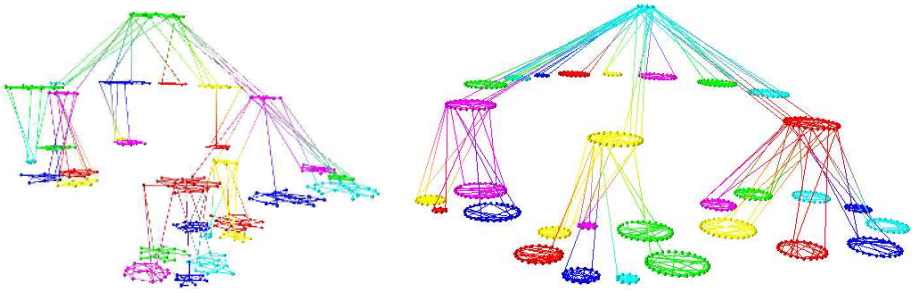

Fig. 3. Cone tree layout (a) a tree with 341 vertices, 1269 edges and 30 clusters drawn with spring layout (b) a tree with 616 vertices, 1636 edges and 31 clusters drawn with circular layout

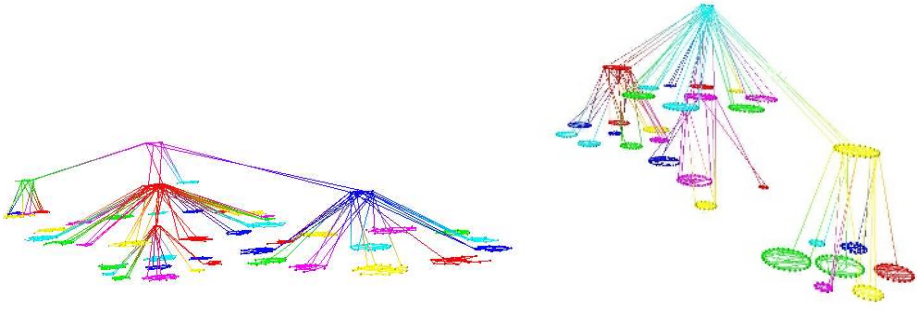

Fig. 4. Rod tree layout (a) a tree with 620 vertices, 1769 edges and 47 clusters drawn with spring layout (b) a tree with 616 vertices, 1636 edges and 47 clusters drawn with circular layout

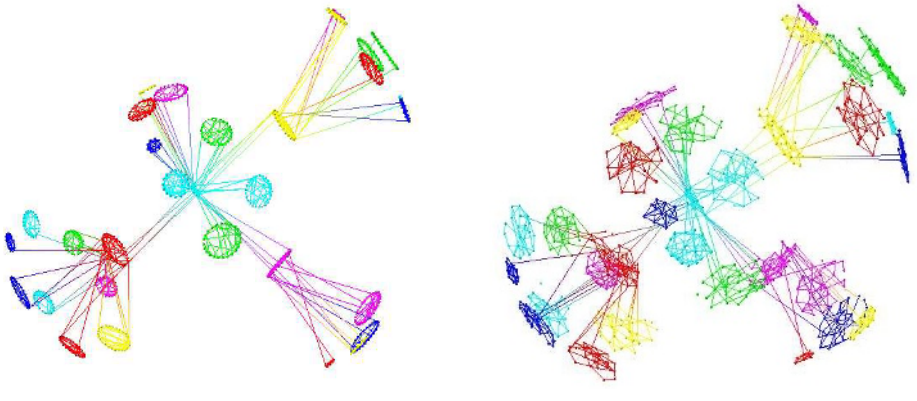

Fig. 5. Free tree layout a tree with 616 vertices, 1636 edges and 31 clusters drawn with (a) circular layout (b) spring layout

network can be divided into many functional units called pathways. Different pathways are connected by sharing the same chemicals. Existing approaches either focus on only visualising individual pathways or the entire network [2, 3, 8].

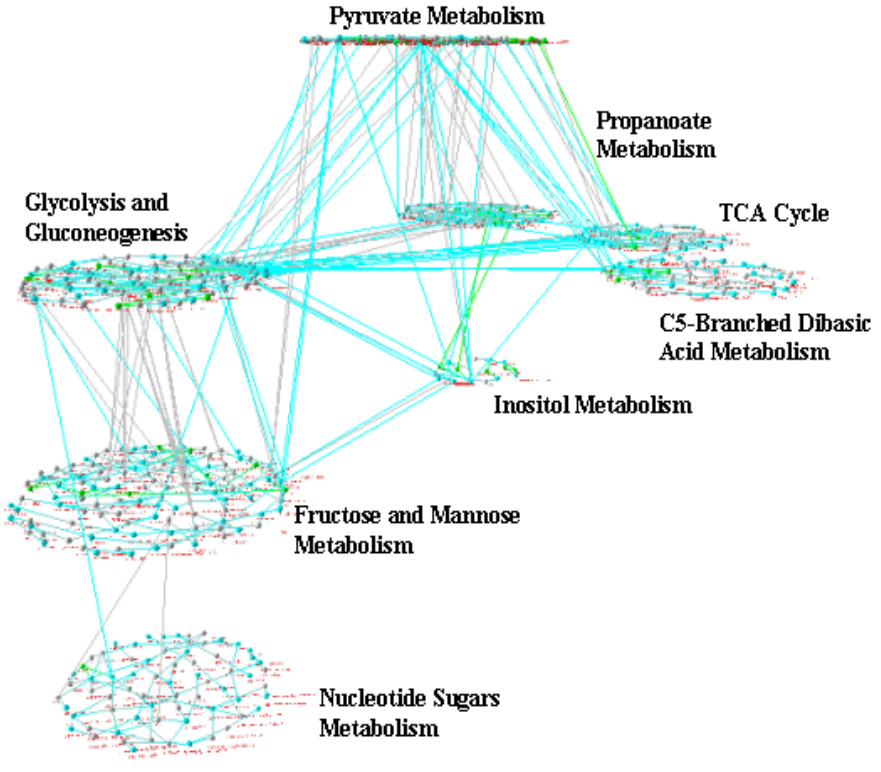

Fig. 6. Metabolic Network of eight related pathways

Clearly, these approaches are not sufficient in conveying the biological functionality of the network. A visualisation method that can effectively display the whole network as well as highlighting the functional independence of the individual pathway is needed.

In order to address this problem, we use the cluster graph model to visualise the metabolic network. Chemicals are represented by nodes and reactions are represented by edges. Each pathway is a cluster, and the whole metabolic network forms a clustered graph. Sharing of chemicals between pathways is represented by inter-cluster edges.

In our study, we use metabolic network data retrieved from the KEGG database (http://www.genome.ad.jp/kegg/). GML [10] files containing individual pathways are read from the Clustered Graph Reader in GEOMI, then the whole network is constructed by adding inter-cluster edges between the same chemicals on different pathways. Since the supergraph of the metabolic network is not a tree, a maximum spanning tree of the original supergraph is used. The supernode with the highest degree centrality is chosen to be the root of the spanning tree.

Figure 6 shows a metabolic network of eight related carbohydrate metabolic pathways. The graph contains 600 nodes and 745 edges. Overall, our approach

in visualising this metabolic network has a number of advantages. Firstly, the functional independence of each pathway is emphasized. Since each pathway is drawn on a different 2D plane, individual pathways can be readily identified, while retaining the overview of how the pathways are connected in the entire network in 3D.

Next, the *central* pathway in the network is identified. For example, both pyruvate metabolism and the glycolysis pathways are connected to most of the other pathways in the network. This means that they may have a significant biological role as the removal of these pathways would essentially disconnect the network. On the other hand, the Nucleotide Sugars Metabolism pathway is quite isolated from the rest of the network.

Thirdly, the relative size of the pathway is easily comparable. For example, the Inosital metabolism pathway is relatively small compared to the other seven similar-sized pathways.

Finally, the connectivity between adjacent pathways is effectively visualised. For example, glycolysis and gluconeogenesis share a lot of chemicals with the TCA cycle as shown by the large number of inter-cluster edges between them. This infers that these two pathways are probably biologically closely related.

4 Conclusion and Future Work

A new method for drawing clustered graph in 3D is presented. The divide-and-conquer algorithm draws each cluster separately in a 2D plane and then combines each drawing of a cluster to construct a drawing of the whole clustered graph in 3D. For this, we designed three linear time weighted tree drawing algorithms in 3D.

We have implemented a series of six different layouts for clustered graphs by combining three 3D tree layouts and two 2D graph layouts. The experimental results show that the resulting drawing can clearly display the structure of the cluster and the relationships between the clusters. Further the use of 2D plane ideas reduces the occlusion problem in 3D and the resulting drawing is easy to navigate.

In the future, different combination of cluster/supergraph layout algorithms can be explored. Application of this clustered graph drawing method can be further explored in the context of other network visualisation applications, for example social networks, communication networks and web-graphs. Further one can design efficient navigation methods for the clustered graph layout for user interaction.

Acknowledgement

The work was supported by a NICTA summer vacation scholarship and an Australian Research Council (ARC) grant. We also thank Dr. Falk Schreiber for providing the metabolic network dataset in GML format.

References

1. A. Ahmed, T. Dwyer, M. Forster, X. Fu, J. Ho, S. Hong, D. Koschützki, C. Murray, N. Nikolov, A. Tarassov, R. Taib and K. Xu, GEOMI: GEOmetry for MAximum Insight, Proceedings of Graph Drawing 2005, 2005.
2. F. Brandenburg, M. Forster, A. Pick, M. Raitner and F. Schreiber, BioPath, Proceedings of Graph Drawing 2001, pp. 455-456, 2001.
3. U. Brandes, T. Dwyer, and F. Schreiber, Visualization of Related Metabolic Pathways in Two and a Half Dimensions, Proceedings of Graph Drawing 2003, pp.111-121, 2003.
4. J. Carriere and R. Kazman, Visualization of Huge Hierarchies: Beyond Cone Trees, Proceedings of IEEE Symposium on Information Visualization 1995, pp. 74-81, 1995.
5. G. Di Battista, P. Eades, R. Tamassia and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice-Hall, 1998.
6. P. Eades and Q. Feng, Multilevel Visualization of Clustered Graphs, Proceedings of Graph Drawing 1996, pp. 101-112, 1996.
7. Y. Frishman and A. Tal, Dynamic Drawing of Clustered Graphs, Proceedings of IEEE Symposium on Information Visualization 2004, pp. 191-198, 2004.
8. B. Genc and U. Dogrusoz, A Constrained, Force-Directed Layout Algorithm for Biological Pathways, Proceedings of Graph Drawing 2003, pp. 314-319, 2003.
9. F. van Ham, H. van de Wetering and J. can Wijk, Visualization of State Transition Graph, Proceedings of IEEE Symposium on Information Visualization 2001, pp. 59-63, 2003.
10. M. Himsolt, GML: Graph Modelling Language. Report, University of Passau, Germany, December 1996. http://www.uni-passau.de/
11. S. Hong and P. Eades, Drawing Trees Symmetrically in Three Dimensions, Algorithmica, 36(2), pp. 153-178, 2003.
12. G. Robertson, J. Mackinkay, and S. Card, Cone Trees: Animated 3D Visualizations of Hierarchical Information, Proceedings of CHI'91, pp. 189-194, 1991.
13. C. Ware and G. Franck, Viewing a Graph in a Virtual Reality Display is Three Times as Good as a 2-D Diagram, IEEE Conference on Visual Languages, pp. 182183, 1994.
14. C. Ware, Designing with a 2 1/2D Attitude, Information Design Journal 10 (3), pp. 171-182, 2001.