A Fully Animated Interactive System for Clustering and Navigating Huge Graphs

Mao Lin Huang and Peter Eades

Department of Computer Science and Software Engineering
The University of Newcastle, NSW 2308, Australia
{mhuang,eades}@cs.newcastle.edu.au

Abstract. This paper describes DA-TU, which combines an animated clustering and an online force-directed animated graph drawing method for the visualization of huge graphs.

1 Introduction

Graphs which arise in Information Visualization applications are typically very large: thousands, or perhaps millions of nodes. Recent graph drawing competitions have shown that visualization systems for classical graphs are limited to (at best) a few hundred nodes.

Attempts to overcome this problem have proceeded in two main directions:

Clustering. Groups of related nodes are "clustered" into super-nodes. The user sees a "summary" of the graph: the super-nodes and super-edges between the super-nodes. Some clusters may be shown in more detail than others. An example is in Figure 1. Note that "New South Wales" is shown in more detail than "Victoria".

Navigation. The user sees only a small subset of the nodes and edges at any one time, and facilities are provided to navigate through the graph.

This paper introduces DA-TU, a system which combines both approaches. The following section briefly describes the model on which DA-TU is built. Some remarks on the implementation of DA-TU, especially with respect to the clustering force model and the animation model, are in Sections 3 and 4; samples of interaction with DA-TU are described in an appendix.

Note: the aim of this paper is to briefly describe the features of DA-TU. The rational behind the design will be described elsewhere. Note that DA-TU is an animated interactive system and it is impossible to fully describe its features on a static page. A video is available from the authors on request.

2 The Framework

The DA-TU system manipulates data in levels, as illustrated in Figure 2. We describe each of these levels, and the functions that operate on them.

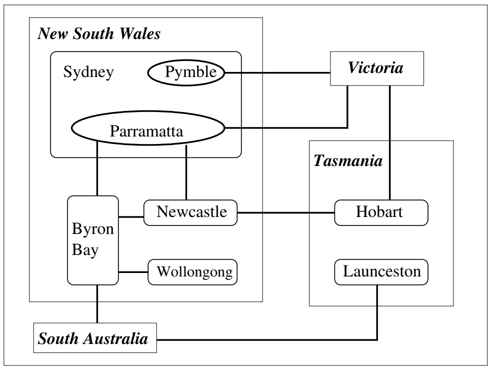

Fig. 1. *Clustered graph.*

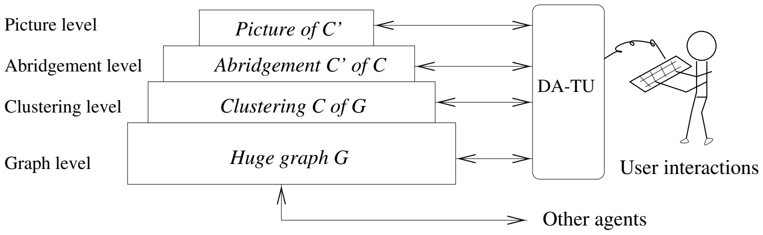

Fig. 2. *The framework of* DA-TU.

2.1 The Graph Level

A graph in DA-TU is a classical undirected graph, consisting of nodes and edges. In applications it is a very large graph, containing many thousands of nodes. The graph may be dynamic, that is, the node and edge set may be changing; these changes may be a result of user interaction through DA-TU, or they may be changed by an outside agent.

2.2 The Clustering Level

A *clustered graph* $C = (G, T)$ consists of an undirected graph $G = (V, E)$ and a rooted tree T such that the leaves of T are exactly the vertices of G [5]. Clustered graphs are closely related to *compound graphs* [9, 8]; however, compound graphs are more general. Each node ν of T represents a *cluster* of vertices of G consisting of the leaves of the subtree rooted at ν. The tree T describes an inclusion relation between clusters; it is the *cluster tree* of C. Figure 1 shows a clustered graph.

DA-TU can operate on a clustered graph $C = (G, T)$ by two basic operations, *create* and *destroy* a cluster. Both can be performed by user interaction, or by an algorithm attached to DA-TU. If a vertex is added to the graph at the graph level, then at the clustering level it is assigned the root of T as a parent; it may be moved to another cluster by the above operations.

2.3 The Abridgement Level

In applications, the whole clustered graph is too large to show on the screen; further, it is too large for the user to comprehend. `DA-TU` draws "abridgements". An abridgement of the clustered graph in Figure 1 is shown in Figure 3. We now

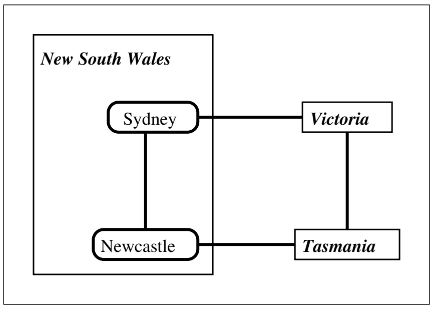

Fig. 3. *An abridgement.*

give a formal definition of "abridgement". Suppose that U is a set of nodes of the cluster tree T. The subtree of T consisting of all nodes and edges on paths between elements of U and the root is called the *ancestor tree* of U. An example of an ancestor tree is in Figure 4.

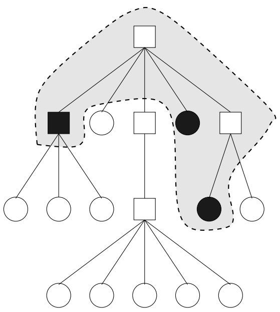

Fig. 4. *The light shaded area is the ancestor tree of the dark shaded nodes.*

A clustered graph $C' = (G', T')$ is an *abridgement* of the clustered graph $C = (G, T)$ if T' is an ancestor tree of T with respect to a set U of nodes of T and there is an edge between two distinct nodes u and v of G' if and only if there is an edge in G between a descendent of u and a descendent of v. Figure 3 shows the abridgement of Figure 1 with basis $\{Sydney\ , Newcastle\ , Tasmania\ , Victoria\}$.

DA-TU has two elementary operations on abridgements; these change the basis of the abridgement. They are *open* a cluster and *close* a cluster.

2.4 The Picture Level

Pictures of clustered graphs are shown in Figures 1, 3, and in Appendix. In this section we define a "picture" of a clustered graph. A *picture* of a clustered graph $C = (G, T)$ contains a location $p(v)$ for each vertex v of G and a route $c(u, v)$ for each edge (u, v) of G, in the same way as drawings for classical graphs. Further, a picture has a region $b(\nu)$ of the plane for each cluster ν of T, such that if ν is a leaf of T then $b(\nu)$ is located at $p(\nu)$, and if μ is a child of ν in T then $b(\mu)$ is contained in $b(\nu)$. The regions currently used by DA-TU are rectangles; we plan to use convex polygons in the next version.

DA-TU provides the usual operation of manually *moving* nodes in a picture. However, the main role of DA-TU is animated automatic *layout*. This is described in the next two sections.

3 The Force Model

In this section, we briefly outline the force model [1, 2, 6, 7]. DA-TU has three types of spring forces:

- **internal-spring:** A spring force between a pair of vertices in the same cluster.
- **external-spring:** A spring force between a pair of vertices in different clusters.
- **virtual-spring:** A spring force between a vertex and a virtual (dummy) node along a virtual (dummy) edge.

It is best to describe these forces with an example; see **Figure 5**. The internal-spring forces on vertex c are along the edges (c, a) and (c, b); the external-spring forces on c are along the edges (c, f) and (c, g).

Virtual-springs can be described using a virtual node in each cluster. In the clusters X, Y, and Z, virtual nodes x', y', and z' are shown; each virtual node is connected to each node in its cluster by a virtual edge. Virtual-springs exert forces along these edges. Note that virtual nodes and edges are not shown in the actual picture of the clustered graph unless the user wants to see them.

As well as spring forces, between each pair of nodes there is a gravitational repulsion force.

The forces are applied additively to give an aesthetically pleasing layout of the graph. The sum of forces on each node is continually computed, and the nodes move according to the strength and direction of these forces. The details of the forces and their implementation is described elsewhere. The force approach is computationally expensive. However, at any one time, DA-TU only deals with a small graph, and there are no problems running on an average PC.

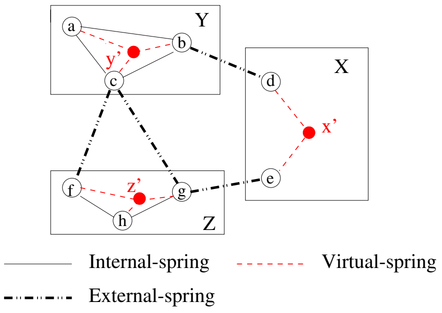

Fig. 5. *Spring forces.*

4 Animations

In DA-TU, the whole visualization is fully animated. Every transition, whether triggered by the user, DA-TU, or by another agent, has its own specific animation. This greatly reduces the cognitive effort of the user in recognizing the new view and change; we aim for a full preservation of the user's "mental map" [3].

More specifically, there are eight types of animation that are implemented in our system. Five of these are specifically related to the clusters: animated gathering, the animation of cluster boundaries, the animation of scaling operations, and animated opening and closing of clusters. Three types of animations are similar methods described in [4]: animated viewing, animated drawing, and animated addition and deletion of nodes and edges.

5 Conclusions

DA-TU provides methods for handling huge graphs visually. The user begins with a picture of an abridgement of the huge graph. At all times, a force-directed animated layout algorithm ensures that the picture is aesthetically pleasing, and that transitions between pictures do not destroy the "mental map" [3]. The Appendix below gives screen dumps from sessions with DA-TU.

References

[1] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Graph drawing - algorithms for geometric representations of graphs. *Prentice-Hall*, 1998 (to appear).
[2] P. Eades. A heuristic for graph drawing. *Congr. Numer.*, 42:149–160, 1984.
[3] P. Eades, W. Lai, K. Misue, and K. Sugiyama. Preserving the mental map of a diagram. In *Proceedings of Compugraphics 91*, pages 24–33, 1991.
[4] Peter Eades, Robert F. Cohen, and Mao Lin Huang. Online animated graph drawing for web navigation. *In G. DiBattista, editor, GD'97, Lecture Notes in Computer Science. Springer-Verlag.*, 1353:330 – 335, 1997.

[5] Qingwen Feng. Algorithms for drawing clustered graphs. In *PhD thesis, Department of Computer Science and Software Engineering, The University of Newcastle, Australia.*, 1997.
[6] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Softw. – Pract. Exp.*, 21(11):1129–1164, 1991.
[7] T. Kamada. Symmetric graph drawing by a spring algorithm and its applications to radial drawing. *Department of Information Science, University of Tokyo*, 1989.
[8] Georg Sander. Layout of compound directed graphs. In *Technical Report A/03/96, University of the Saarlands*, 1996.
[9] K. Sugiyama and K. Misue. Visualization of structural information: Automatic drawing of compound digraphs. *IEEE Transactions on Systems, Man and Cybernetics*, 21(4):876–896, 1991.

Appendix: Examples

This section contains two sequences of screen dumps from DA-TU. They illustrate the basic operations of the system and how it works to achieve a better quality of the layout of clustered graphs. The animation which is an essential feature of DA-TU is lost in these static pictures; a video is available from the authors.

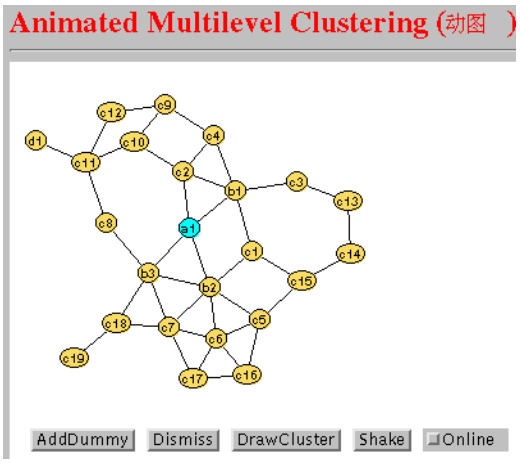

Fig. 6. A graph in DA-TU.

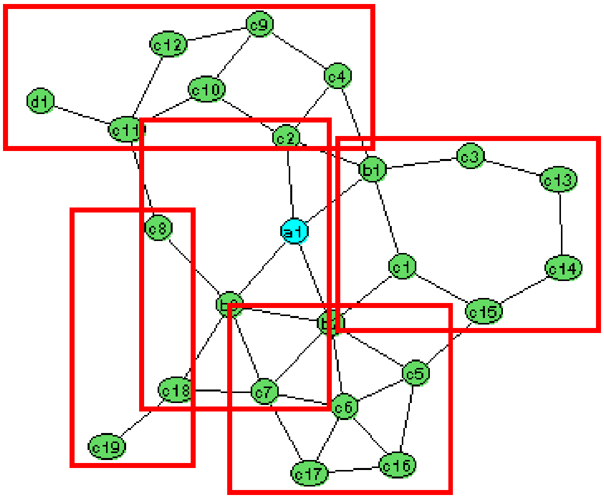

Fig. 7. The user creates five new clusters on the graph in **Figure 6**. However, this layout has five overlaps among the clusters.

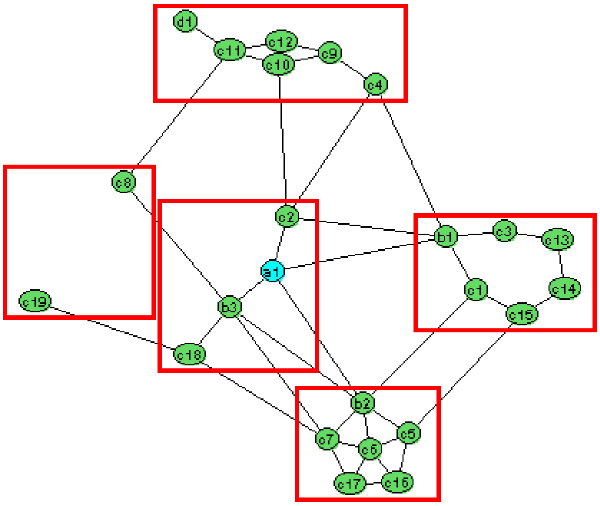

Fig. 8. The user applies the spring forces in the "gathering" operation to eliminate the overlaps in **Figure 7**. This greatly improves the readability of the layout for the user.

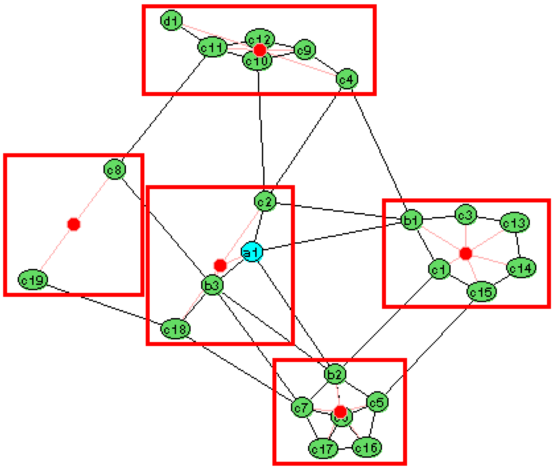

Fig. 9. The same layout as shown in **Figure 8**; however, the virtual nodes and edges are shown. Note the "virtual spring" force applied between non-adjacent vertices $c8$ and $c19$.

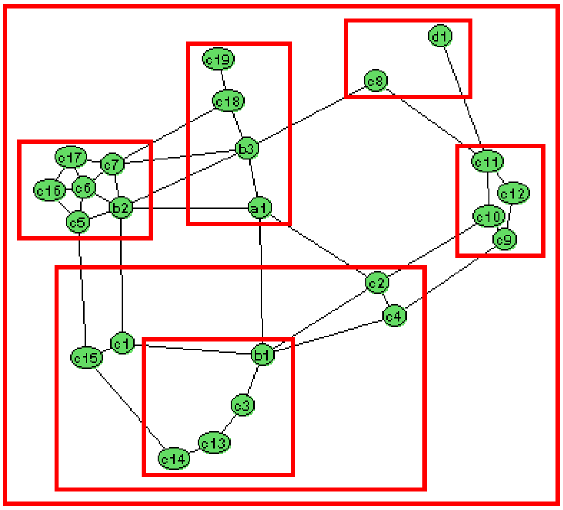

Fig. 10. A clustered graph C with 4 levels, after the application of spring forces in the "gathering" mode.

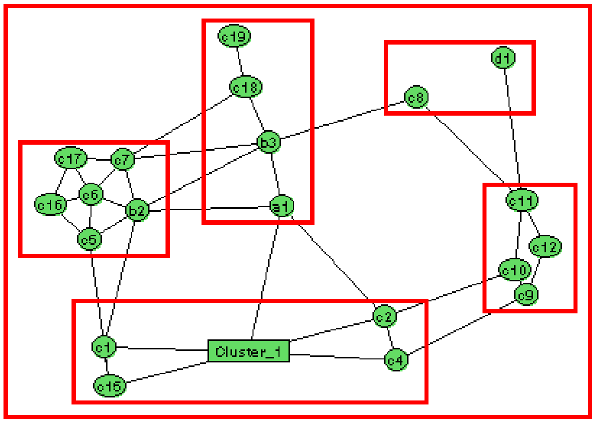

Fig. 11. Closing one cluster, $\nu = \{b1, c3, c13, c14\}$, in **Figure 10** by clicking on ν in the "close" mode. The representation of closed ν is a small rectangle.

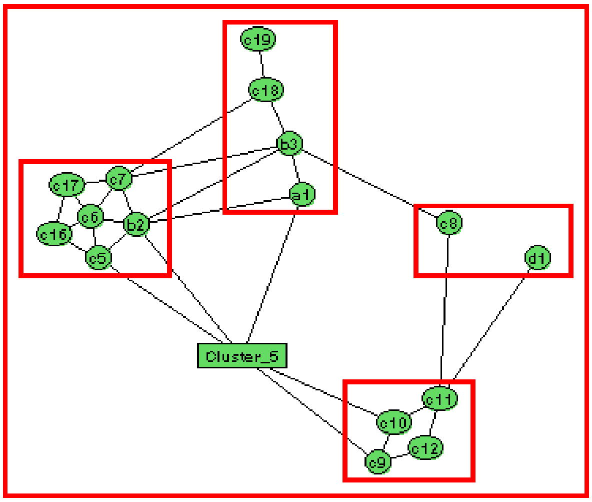

Fig. 12. Continuing, closing the cluster $\{c1, c2, c4, c15, Cluster\_1\}$ in **Figure 11**.

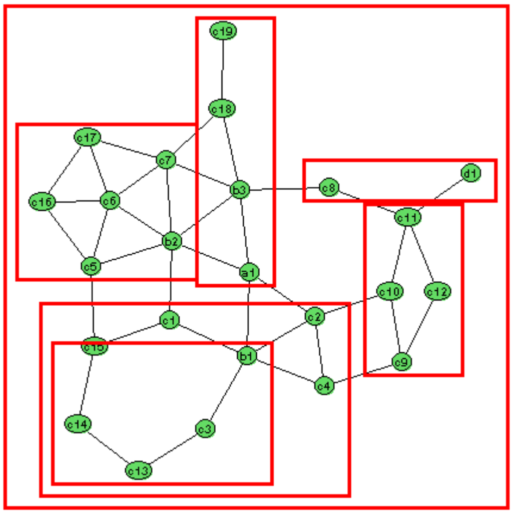

Fig. 13. Open the cluster *Cluster*_5 in **Figure 12**, and then open the cluster *Cluster*_1. This is done by clicking on both visual rectangles of two clusters one by one under in the "open" mode.

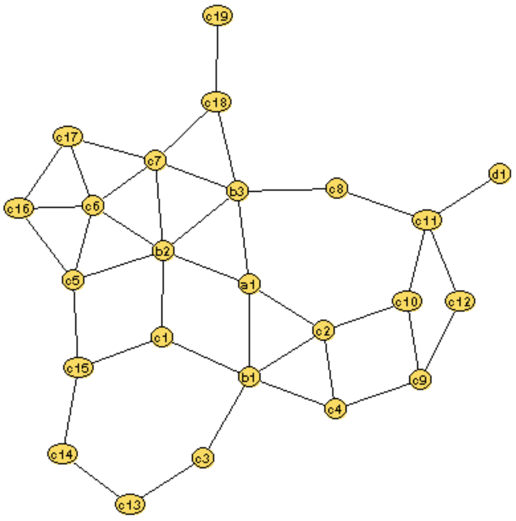

Fig. 14. Dismissing the whole cluster tree from the layout as shown in **Figure 13**, and returning to the graph with no clusters. This could be done by destroying the clusters one by one; however, a "dismiss" button is provide to destroy all clusters at once.