FADE: Graph drawing, clustering and visual abstraction

Aaron Quigley and Peter Eades

http://www.cs.newcastle.edu.au/~aquigley
Department of Computer Science and Software Engineering,
Univ. of Newcastle, Callaghan, NSW 2308, Australia

Abstract. A fast algorithm(FADE) for the 2D drawing, geometric clustering and multilevel viewing of large undirected graphs is presented. The algorithm is an extension of the Barnes-Hut hierarchical space decomposition method, which includes edges and multilevel visual abstraction. Compared to the original force directed algorithm, the time overhead is $O(e + n \log n)$ where n and e are the numbers of nodes and edges. The improvement is possible since the decomposition tree provides a systematic way to determine the degree of *closeness* between nodes without explicitly calculating the distance between each node. Different types of regular decomposition trees are introduced. The decomposition tree also represents a hierarchical clustering of the nodes, which improves in a graph theoretic sense as the graph drawing approaches a lower energy state. Finally, the decomposition tree provides a mechanism to view the hierarchical clustering on various levels of abstraction. Larger graphs can be represented more concisely, on a higher level of abstraction, with fewer graphics on screen.

1 Introduction

Force-directed algorithms are often used for graph drawing due to their flexibility, ease of implementation and their often pleasant resultant drawings. Numerous graph drawing systems have been developed based on such force directed algorithms [9, 36, 8, 5]. The flexibility of the force directed approach allows for many kinds of constraints [16, 3, 31, 13]. However, one common feature of most force directed methods, is their inability to scale to handle large graphs. The problem stems from the high computational cost of computing a force between every pair of nodes.

This paper introduces a fast algorithm for the drawing of large undirected graphs based on the force directed approach. This algorithm hierarchical *groups* nodes. The nonedge force on an individual node from other nodes close by is, on average, evaluated by direct node-to node interaction, whereas the force due to more distant nodes is included as a *group* contribution. The reduction in the computational cost of the force directed approach allows larger graphs to be drawn in real-time.

The fact that larger graphs can be drawn using this method is not enough for visualization. It is easy to draw a great deal of graphical information but the limits on both the available screen space and more importantly human cognition are quickly reached [35, 34, 33, 20].

The rest of this paper is organized as follows. In section 2, we review the concepts of *Graph Theoretic Clustering* and *Geometric Clustering* and how they relate to the visualization of large graphs. In section 3 we outline *hierarchical space decomposition* and *tree codes*. Our algorithm FADE is described in section 4. Section 5 describes the *progressive cycle* of clustering, based on a step-wise improvement in the drawing. We discuss the generation of the *visual abstraction* in section 6. And finally, in section 7 our results for performance, error rates and clustering are presented.

2 Clustering

2.1 Graph Theoretic Clustering

A *clustering* of a graph, $G = (V, E)$ consists of a partition $V = V_1 \cup V_2 \cup ....\cup V_k$ of the node set of G. *Graph theoretic clustering* is the process of forming clusters based on the structure of the graph [22, 29, 23, 6, 24, 30]. The usual aim is to form clusters that exhibit a high *cohesiveness* and a low *coupling*; these may be measured, for example, by the number of *intra*-cluster edges and the number of *inter*-cluster edges respectively. For example, the graph with the adjacency table in Figure 1 has been divided into two clusters with only one edge coupling them. Methods for graph theoretic clustering, motivated by applications in parallel computing, data mining, and VLSI design are well developed [6, 24, 30].

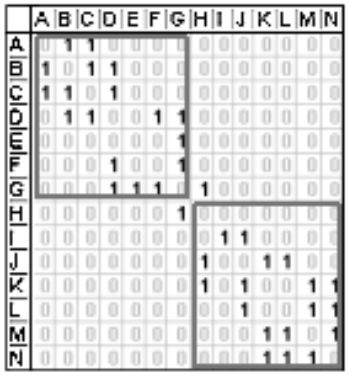

Fig. 1. Graph Theoretic, single linkage clustering example

One of the main problems for graph theoretic clustering methods is selecting a threshold of coupling, without some *a priori* knowledge of the data. A low

threshold results in too many clusters with too few elements. A high threshold results in large tree-like clusters (*dendograms*), with the degenerate case of all the nodes collapsing into a single cluster.

2.2 Clustered Graphs

A clustered graph is a graph with a recursive clustering, or partitioning, of the node set of G [26]. Graph drawing algorithms have been developed to draw clustered graphs [37, 19, 27, 7, 26]. Recently, methods have been developed that exploit clusters to improve the performance of the underlying drawing algorithm. Harel et al. [11, 10] show how a force directed method can use the graph theoretic structure to approximate long range force interactions; here *node to cluster* forces are computed rather than *node to node* forces. However, approximating forces based on the graph theoretic structure has some drawbacks:

1. The computational cost in discovering the recursive clustering can quickly dominate as the graphs become large.
2. If the graph theoretic clustering is poor, then there is no time saving from such an approximate force scheme.
3. The resultant graph drawing can be distorted, based on the clustering scheme chosen; this may adversely affect its usefulness in visualization.

The partitioning scheme chosen in [11] may be improved by the use of multi-level partitioning schemes [21, 22] where the graph is coarsened, then the smaller graph is partitioned, and then uncoarsened to construct a partitioning of the original graph. It should be noted, that the scheme introduced in [11], produces nice drawings of large graphs that exhibit good graph theoretic clusters. For graphs, such as trees, it is noted that a post-processing *beautification* step, on the entire graph, must be performed.

2.3 Geometric Clustering

A *geometric clustering* of a set S of points in k-dimensional space is a partition of $S = S_1 \cup S_2 \cup .... \cup S_p$ into subsets. The usual aim of a geometric clustering method is to have points which are close to each other sharing a cluster and points that are far apart in different clusters. Geometric clustering is used in a wide variety of data mining applications; the k attributes of an entity are mapped to a point in k-dimensional space, and an appropriate distance function is used [32, 6, 24, 30]. One such function is the *K-means algorithm*, which is based on a distance metric where the attributes are scaled so that the Euclidean distance between sites is appropriate [12].

In many circumstances, a geometric clustering is defined by a *space decomposition*, such as the Voronoi diagram in Figure 2. Geometric clustering may be applied to the location of nodes in a graph drawing; however, unlike graph theoretic clustering, the edges are not used to determine the geometric clustering.

2.4 Graph Drawing and Clustering

A Geometric clustering method [32, 6, 24] applied to a graph drawing induces a graph theoretic clustering, but it may have low quality in terms of coupling and cohesion [21]. To ensure that the graph theoretic clustering has high quality, the graph drawing must exhibit a strong relationship between geometric and graph theoretic distance between nodes.

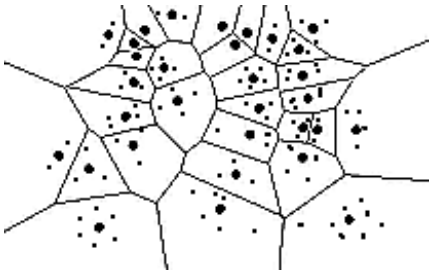

Fig. 2. Voronoi diagram based on pseudnodes, real nodes shown as points

Our graph drawing method (FADE) uses a fast recursive space decomposition, which induces a geometric clustering of the locations of the nodes; this in turn induces a graph theoretic clustering. This graph theoretic clustering is then used in a force directed algorithm that improves the relation between Euclidean and graph theoretic distances in the drawing, and this in turn improves the graph theoretic clustering. Iterating this process improves both the drawing and the clustering. Viewing the graph drawing on different abstraction levels, of the cluster tree, gives a set A=A_1,A_2,A_3...A_x of abstractions, where x is the height of the decomposition tree.

3 Hierarchical space decomposition

Modeling large numbers of interacting particles has interested physicists for centuries. Typical examples include celestial mechanics, plasma simulations and the vortex method in fluid dynamics [25, 14, 18, 15]. Such numerical simulations are referred to as the *N-body problem* and typically involve 10^{12} - 10^{23} particles. In classical systems, the *potential* (or force) has the form:

$$\phi = \phi_{short} + \phi_{long} + \phi_{external}, \tag{1}$$

where ϕ_{short} is a rapidly decaying function of distance (such are the Van der Waals potential in chemical physics), $\phi_{external}$ is a function which is independent of the number and relative positions of the particles (such as a magnetic field); these forces can be computed in time O(N). However, the computation of ϕ_{long}, the particle to particle interactions, takes time O(N^2). The ϕ_{long} force can be

thought of as an *electrical repulsion* force in the *classical force-directed* methods (Chapter 10 [9]) for graph drawing. The cost of $O(N^2)$ is excessive in most cases, and prohibitive in others. When using the force-directed approach for graph drawing the same limitations apply. In Physics different approaches have been developed to reduce the computational effort in calculating the long range ϕ_{long} forces by means of *approximation* or *estimation*.

One such method is called *particle-in-cell (PIC)*: a regular grid is laid over the simulation area and the particles contribute their masses to create a source density. These source densities are used in the estimation of the forces on a particular particle. PIC codes have been popular in particle simulations but have difficulties dealing with non-uniform particle distributions, that is, where the particle distribution is clustered [25]. The modified force directed algorithm of Fruchterman and Reingold [8] follows the PIC approach. PIC based methods prove unsuitable for approximating the forces in graph drawings which do not exhibit a uniform node distribution.

3.1 Tree Codes

A systematic and recursive division of space into a series of *cells*, called a *tree code* in the astrophysics literature [2, 25, 18, 4, 1], can be used to speed up n-body force calculations. The force on an individual particle from other particles close by is, on average, evaluated by direct particle-to-particle interaction, whereas the force due to more distant particles is included as a group contribution. Tree codes were first developed in the context of galactic simulations involving hundreds of billions of bodies. Tree codes may be based on regular (*Quad-tree, Oct-tree*) and irregular (*Voronoi*) space decompositions. A *nono-tree* (a nine-way recursive decomposition), is illustrated in Figure 3. Regular trees such as this can be built in linear time.

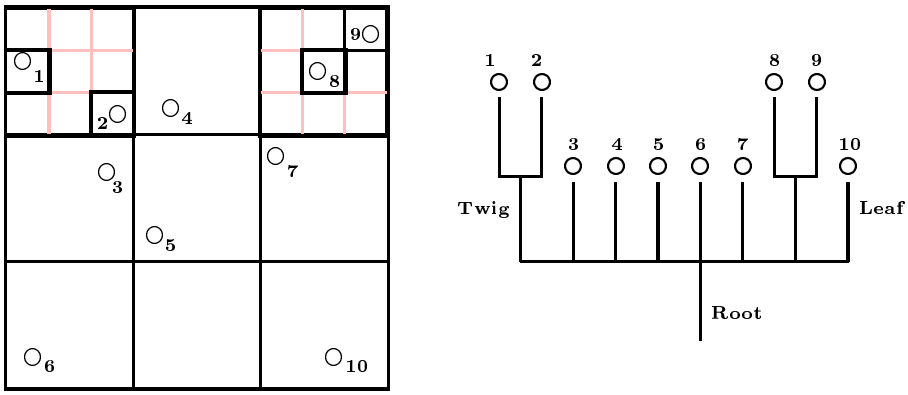

Fig. 3. Nono-tree space decomposition and data structure

Unlike PIC codes building a decomposition tree does not suffer from a non-uniform distribution of particles.

3.2 Tree-codes and clustering

The decomposition of the nodes of a graph drawing based on a tree-code method generates a recursive geometric clustering of the vertices of the graph. The recursive clustering, represented as sub-trees, does not directly produce a high *quality* geometric clustering of the node positions, nor is it necessarily a high quality graph theoretic clustering. However the clustering does facilitate an improvement in the performance of force directed algorithms, as in shown in Section 4.1 below. Further, it allows for multi-level viewing of huge graphs at various levels of abstraction. Finally, as the quality of the drawing improves (that is, it reaches a lower energy state of the force system), the quality of suggested clustering, exhibited by the decomposition tree, improves.

4 FADE algorithm

The main idea of the FADE algorithm is the following. Using the decomposition tree the *nonedge force* on an individual node from other nodes close by is, on average, evaluated by direct node-to-node interaction, whereas the force due to more distant nodes is included as a node cluster contribution. The main difference with the graph theoretic cluster approach of Harel et al. [11, 10] is that we approximate forces based on actual geometric information, rather than preprocessed approximate forces. This difference allows our force directed method to be applied to large scale graphs, without distorting the resultant drawing due to the underlying graph structure.

4.1 Implementation

We have implemented FADE, a drawing algorithm for undirected graphs, in two dimensions, using a simple force model and a quad-tree space decomposition. Force directed algorithms, such as those in [16, 3, 13, 9, 36, 8] often have two kinds of forces: *edge-forces* such as Hooke's law spring forces, and *non-edge forces*, such as magnetic repulsion. Typical graphs in graph drawing applications have $O(n)$ edges and $\Omega(n^2)$ non-edges, where n is the number of nodes. The FADE algorithm computes node-node forces for edges, and approximates non-edges forces using a tree-code approach. Where the tree is reconstructed *ab initio* for every new iteration of the force directed algorithm FADE.

Roughly speaking, FADE works as follows:

```
REPEAT:
  Construct geometric clustering using a space decomposition
  Compute edge forces
  Compute nonedge forces (node-node and node-pseduonode)
  Move nodes
UNTIL convergence
```

The tree structure, provided by a recursive space decomposition such as a quad-tree, provides a systematic way to determine the degree of *closeness* between nodes without explicitly calculating the distance between each node, see for example Figure 8. The nonedge force between near nodes is calculated directly whereas more distant nodes are grouped together into super-nodes or pseudonodes.

The most basic means for determining closeness is a *tolerance criterion* [25, 2]. We test $s/d \leq \theta$, where s is the width of the cell, d is the distance between the current node and the center of mass of the cell and θ is the fixed-tolerance parameter. If $s/d \leq \theta$, then the internal nodes are ignored and the force contribution of the pseudonode is added to the cumulative force for that node. Otherwise, the pseudonode is resolved into its daughter pseudonodes (sub-trees), each of which is recursively examined. Pseudonodes are resolved by continuing the decent through the tree until either the tolerance criterion is satisfied or a leaf node is reached.

Using a quad-tree space decomposition and a value of 1.0 for θ the fixed-tolerance parameter, some examples are given for the node set shown on the left of Figure 4.

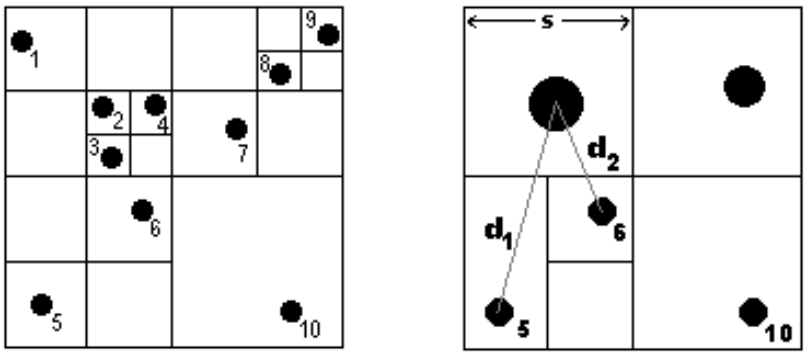

Fig. 4. Comparing node 5 and 6 with the North-West pseudonode, where s/d=0.80 and s/d=1.46 respectively

In Figure 4, the leaf node 5 is compared with the pseudo particle in the North-West quad. The weight of the pseudonode is the cumulative weight of the leaves in its sub-tree, in this case 4. The s value is the width of the cell the pseudonode covers and d_1 is the distance between the node and the pseudonode. $\frac{s}{d} = 0.80$ in this case. Based on the value 1.0 for θ the FADE algorithm computes the nonedge force between this pseudonode and node 5, it then adds this force to the cumulative nonedge force for node 5.

Leaf node 6, is compared with the pseudo particle and d_2 is the distance between the node and the pseudonode in the North-West quad to get, $\frac{s}{d} = 1.46$. Here the value does not fulfill the criterion $s/d \leq \theta$, hence the pseudonode is re-

solved into its sub-nodes (in this case leaf-node 1 and the pseudonode representing 2,3,4). The algorithm continues by testing $s/d \leq \theta$ for the pseudonode(2,3,4) and if it fulfills the criterion then a node to pseudonode force calculation is performed. The *interaction list* for a given node, is the list of pseudonodes (twig nodes) and nodes (leaf nodes) involved in the calculation of its nonedge force.

For nonzero θ the time required to compute the force in a single iteration of FADE scales as $O(n \log n)$ [4, 14]. The actual interaction list for a given node depends strongly on the choice of the tolerance parameter θ. The case $\theta = 0$ is equivalent to computing all node-node nonedge forces, which is the classical approach in $O(N^2)$ force-directed methods. However a very large θ results in the calculation of only forces between nodes and very large pseudonodes. It would be very quick, approaching $O(N)$ but very inaccurate. The high level of inaccuracy exhibits itself as a poor drawing of the graph, based on a force computation.

There is a cross-over point in the number of nodes where hierarchical tree codes becomes more efficient than the direct method. Optimizing the tree construction and the force calculations by vectorisation or parallelization lowers this cross-over point. When the number of dimensions in the force law is low this can be in the order of a few hundred to several thousand nodes [25].

5 Progressive Cycle

The space decomposition tree generated at each timestep in the FADE algorithm represents a recursive geometric clustering of the nodes of the graph. The drawing typically moves to a lower energy state from timestep to timestep. This is as a result of the edge and nonedge forces. Overall, the drawing improves as nodes with an edge between them, are on average drawn closer than nodes without. The geometric clustering generated also improves as the drawing improves; this process in referred to as the *progressive cycle*. Various geometric measures of clustering can be applied to each recursive clustering, and this allows for a measure of how the clustering improves as the forces are applied.

Results from such a measure can be seen in figure 5, the measure here is the number of *inter-cluster edges* vs *intra-cluster* edges. Here we show this measure applied to different levels in the hierarchical geometric clustering over 100 iterations of *fade* for a graph of 1020 nodes. This measure shows the result of the drawing reaching a stable state, which induces an improvement in the quality of the clustering over time, measured in terms of the coupling.

6 Visual Abstraction

If the drawing of the graph cannot easily be viewed on screen, for example due to poor resolution, or the sheer volume of graphical information, then the visualization can no longer be considered useful to the user. Numerous viewing schemes have been proposed to overcome this problem [17, 9, 28, 16, 7]. FADE follows the clustered graph drawing approach. It has been shown, that the tree built by the FADE algorithm produces a hierarchical clustering of the vertices of

the graph. Similar to other cluster graph drawing techniques, a particular level in the cluster tree (an *abridgement*) can be drawn on screen. The choice of level allows the graph to be concisely (i.e. more abstractly) represented with fewer graphics on screen.

The abridgement of the graph drawing results in a loss of precision. This has to be measured against the fact that a greater understanding of the overall graph is now possible. A visualization scheme based on a multi-level viewing still has the ability to mix levels of abstraction, allowing greater detail in certain areas of interest, while maintaining an abstract view of the rest of the graph drawing. The amount of graphical elements presented to the user in Figure 6(left side) (16 pseudonodes and 33 edges) is approximately 0.67% of that displayed in drawing of the underlying graph on the right side in Figure 7 (2500 nodes and 4900 edges). The first drawing can be easily be drawn in realtime and it still provides a visual approximation to the overall shape of the graph, and if varying pseudonode sizes were used, to the node distribution. Several examples of multi-level viewing, based on a regular recursive decomposition of space, are shown in the Appendix.

7 Experimental Results

Nodes	Direct (sec)	FADE (sec)	% Error
512	0.455	0.04	0.513
1020	1.82	0.088	0.592
1442	3.61	0.168	0.675
2500	10.88	0.202	0.622
6000	62.66	0.676	0.673
10510	192	1.704	0.449
22800	920	3.36	0.561
30000	1593	3.546	0.517
40960	2979	5.592	0.567
49284	4316	6.730	0.628
105233	19604	13.371	0.481

Table 1. Experimental Comparison of tree-code Vs direct force calculation

We ran our experiment based on a Parlance implementation running on a Pentium III 500 Mhz processor. The performance results were as expected, from the physics literature [4, 25], with a large improvement over the classical direct approach per iteration. The error rate is a vector measure computed from the direct non-edge forces and the approximate non-edge forces computed in the FADE algorithm. It is important to note the error rates, which are less than 1%, do not grow as the graphs get larger. The error rate is further damped when the edge forces are considered.

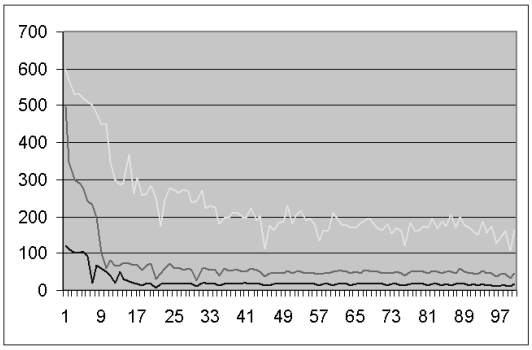

Fig. 5. Cluster measure for levels 4, 6, 8 in the clustering

8 Conclusions and future work

FADE is an algorithm for the drawing and multi-level visualization of large undirected graphs. It is based on an extension of tree-codes for the N-body problem in physics. The algorithm is effective in drawing large graphs and the use of the space decomposition tree provides a new and novel method for the multi-level viewing of large graphs and provides a geometric clustering of the nodes of the graph. Future work will investigate the use of other quality measures for graph clustering and what affects other regular and irregular space decompositions have on these measures and on the visual abstraction.

Finally, approaches to data mining based on the generation of clusters from a hierarchical space decomposition have been developed [39, 38]. These methods analyze weights of sub-trees to discover candidate clusters and then compose adjacent sub-trees, based on various measures, to form the clusters. The decisions are based on the weights of the sub-trees, which are inherently localized. For geometric graph cluster analysis, it should be possible, to integrate the force model into the decision process, thereby taking global information into account. Not only the size of the sub-tree but also how often this sub-tree is considered to be a pseudonode in the force computation.

References

1. Richard J. Anderson, *Tree data structures and n-body simulation*, SIAM J. Comput. **28** (1999), no. 6, 1923–1940.
2. J. Barnes and P. Hut, *A hierarchical o(n log n) force-calculation algorithm*, Nature **324** (1986), no. 4, 446–449.
3. Francois Bertault, *A force-directed algorithm that preserves edge crossing properties*, Seventh International Symposium on Graph Drawing (Prague, Chezh Republic) (Jan Kratochvíl, ed.), Springer, September 1999, pp. 351–358.
4. G. Blellcoh and G. Narlikar, *A practical comparison of n-body algorithms*, Technical report, Wright Laboratory, 1991.

5. Peter Eades, *A heuristic for graph drawing*, Congresses Numerantium **42** (1984), 149–160.
6. G. Erhard, *Advances in system analysis vol. 4, graphs as structural models: The application of graphs and multigraphs in cluster analysis*, Vieweg, 1988.
7. Q.W. Feng., *Algorithms for drawing clustered graphs*, Ph.D. thesis, The University of Newcastle, Australia, 1997.
8. T. Fruchterman and E. Reingold, *Graph drawing by force-directed placement*, Software-Practice and Experience **21** (1991), no. 11, 1129–1164.
9. Roberto Tamassia G. Di Battista, P. Eades and I. G. Tollis, *Graph drawing, algorithms for the visualization of graphs*, Prentice-Hall Inc., 1999.
10. R. Hadany and David Harel, *A multi-scale method for drawing graphs nicely*, 25th International Workshop on Graph-Theoretic Concepts in Computer Science, 1999.
11. David Harel and Yehuda Koren, *A fast multi-scale method for drawing large graphs*, Technical report, Dept. of Applied Mathematics and Computer Science, Weizmann Institute, Rehovot, Israel, November 1999.
12. J. Hartigan, *Clustering algorithms*, Wiley, 1975.
13. W. He and K. Marriott, *Constrained graph layout*, Symposium on Graph Drawing, GD '96 (Stephen North, ed.), vol. 1190 of Lecture notes in Computer Science, Springer, 1996, pp. 217–232.
14. L. Hernquist, *Hierarchical n-body methods*, Computational Physics Communications **48** (1988), 107–115.
15. R. W. Hockney and P. M. Sloot, *Computer simulations using particles*, McGraw-Hill, 1981.
16. Mao Huang, *On-line animated visualization of huge graphs*, Ph.D. thesis, The University of Newcastle, Australia, 1999.
17. Maurice M. de Ruiter Ivan Herman, Guy Melançon and Maylis Delest, *Latour - a tree visualisation system*, Seventh International Symposium on Graph Drawing (Prague, Chezh Republic) (Jan Kratochvíl, ed.), Springer, September 1999, pp. 392–399.
18. L. Greengard J. Carriert and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations*, SIAM Journal on Scientific Computing **9** (1988), no. 4, 669–686.
19. Arunabha Sen Jubin Edachery and Franz J. Brandenburg, *Graph clustering using distance-k cliques*, 7th International Symposium on Graph Drawing, GD '99 (Jan Kratochvíl, ed.), vol. 1731 of Lecture notes in Computer Science, Springer, 1999, pp. 98–106.
20. P. Eades et al K. Misue, *Layout adjustment and the mental map*, Journal of Visual Languages and Computing (1995), 183–210.
21. George Karypis and Vipin Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM Journal on Scientific Computing (1998).
22. ———, *A parallel algorithm for multilevel graph partitioning and sparse matrix ordering*, Journal of Parallel and Distributed Computing (1998).
23. B. W. Kernighan and S. Lin, *An efficient heuristic procedure for partitioning graphs*, The Bell System Techinical Journal (1970).
24. M. Lorr, *Cluster analysis for social scientists*, Jossey-Bass Ltd., 1987.
25. Susanne Pfalzner and Paul Gibbon, *Many-body tree methods in physics*, Cambridge University Press, 1996.
26. R. Cohen Q. W. Feng and P. Eades, *How to draw a planer clustered graph*, COCOON, vol. 959, Lecture notes in Computer Science, Springer, 1995, pp. 21–31.

27. Reinhard Sablowski and Arne Frick, *Automatic graph clustering (system demonstration)*, Symposium on Graph Drawing, GD '96 (Stephen North, ed.), vol. 1190 of Lecture notes in Computer Science, Springer, 1996, pp. 395–400.
28. Manojit Sarkar and Marc Brown, *Graphical fisheye views of graphs*, ACM SIGCHI '92 Conference on Human Factors in Computing Systems, March 1992.
29. Horst Simon and Shang-Hua Teng, *How good is recursive bisection*, Nasa - publication, Ames Research Center NASA, June 1993.
30. H. Spath, *Cluster analysis algorithms for data reduction and classification of objects*, Horwood, 1980.
31. K. Sugiyama and K. Misue, *A simple and unified method for drawing graphs: Magnetic-spring algorithm*, Proceedings of Graph Drawing, GD '94 (Roberto Tamassia and Ioannis Tollis, eds.), vol. 894 of Lecture Notes in Computer Science, Springer, 1995, pp. 364–375.
32. S. Teng, *Points, spheres, and separators: A unified geometric approach to graph partitioning*, Ph.D. thesis, School of Computer Science, Carnegir Mellon University, 1991.
33. Edward R. Tufte, *The visual display of quatitative information*, Graphics Press, 1983.
34. ———, *Envisioning information*, Graphics Press, 1990.
35. ———, *Visual explanations*, Graphics Press, 1997.
36. Daniel Tunkelang, *Jiggle: Java interactive general graph layout environment*, Sixth International Symposium on Graph Drawing (McGill University, Canada) (Sue Whitesides, ed.), Springer, August 1998.
37. Andrej Mrvar Vladimir Batagelj and Matjaž Zaveršnik, *Partitioning approach to visualization of large graphs*, 7th International Symposium on Graph Drawing, GD '99 (Jan Kratochvíl, ed.), vol. 1731 of Lecture notes in Computer Science, Springer, 1999, pp. 90–97.
38. Jiong Yang Wei Wang and Richard Muntz, *Sting: A statistical information grid approach to spatial data mining*, 23rd International Conference on Very Large Data Bases(VLDB), IEEE, 1997, pp. 186–195.
39. ———, *Sting+: An approach to active spatial data mining*, IEEE, 1999, pp. 116–125.

9 Appendix

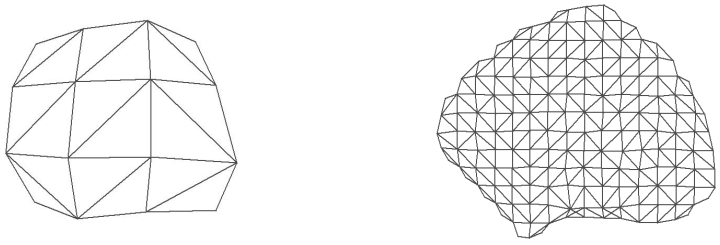

Fig. 6. Graph of 2500 nodes shown on levels 3 and 5 of the decomposition tree

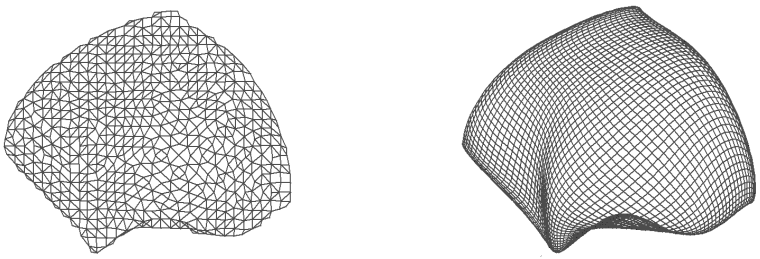

Fig. 7. Graph of 2500 nodes shown on level 6 and the lowest level of the decomposition tree

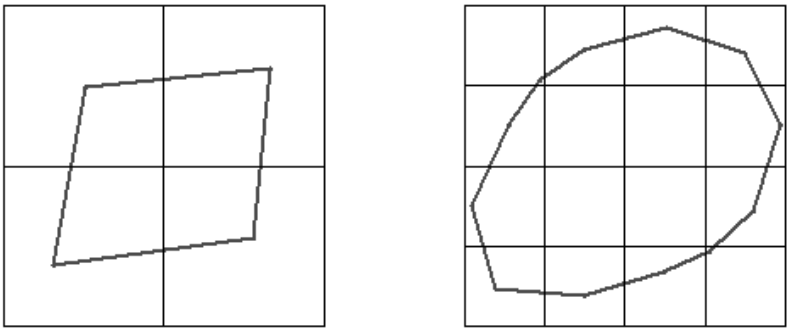

Fig. 8. Cycle of 400 nodes shown on levels 2 and 3 with the space decomposition overlaid

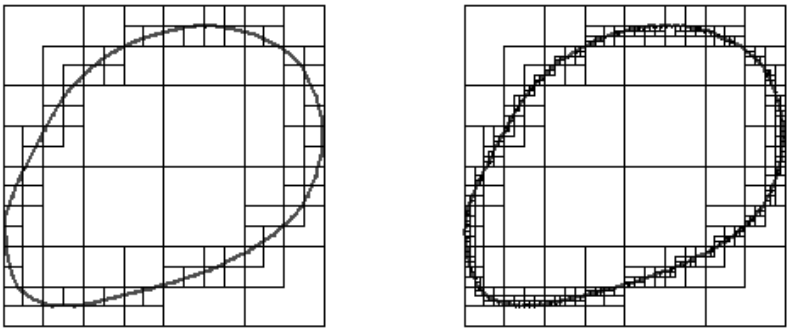

Fig. 9. Cycle of 400 nodes shown on level 5 and the lowest level, with the space decomposition overlaid

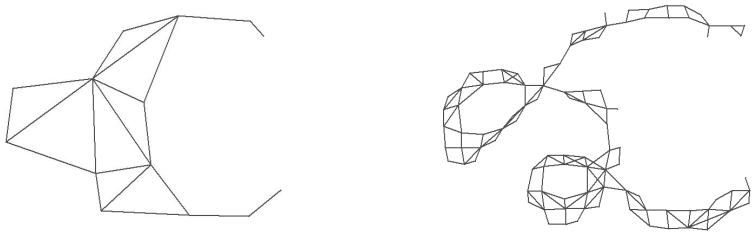

Fig. 10. Graph of 4700 nodes shown on levels 3 and 5 of the decomposition tree

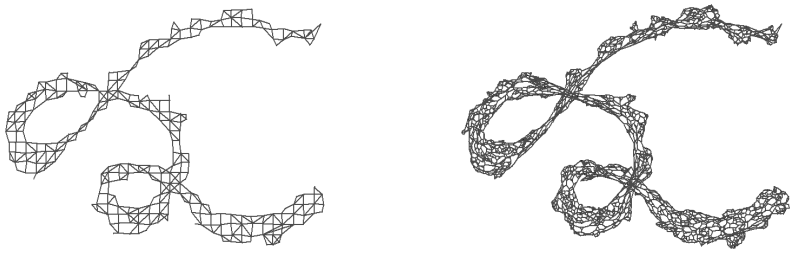

Fig. 11. Graph of 4700 nodes shown on level 8 and the lowest level of the decomposition tree

Fig. 12. Graph of 1400 nodes shown on levels 4 and 6 of the decomposition tree

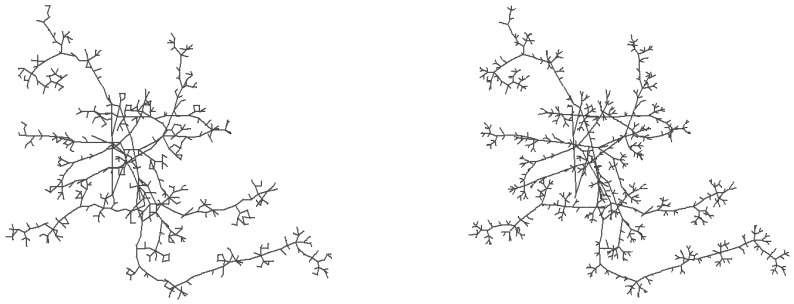

Fig. 13. Graph of 1400 nodes shown on level 8 and the lowest level of the decomposition tree