Recursive Space Decompositions in Force-Directed Graph Drawing Algorithms

K. J. Pulo

Basser Department of Computer Science,
University of Sydney, NSW, Australia, 2006.
Email: kev@cs.usyd.edu.au

Abstract

Force-directed graph drawing algorithms are a popular method of drawing graphs, but poor scalability makes them unsuitable for drawing large graphs. The *FADE paradigm* uses the proximity information in *recursive space decompositions* to address this problem and that of high visual complexity. The FADE paradigm has been presented with a simple and common recursive space decomposition known as the *quadtree*. However, quadtrees have the disadvantage of not being robust with respect to small perturbations and some transformations of the input data, and this can adversely affect the resultant graph drawing. This paper investigates the FADE paradigm using an alternative recursive space decomposition known as the *recursive voronoi diagram*, which avoids some of the problems found in quadtrees at an additional time complexity cost. Preliminary results with random graphs and graphs in the domain of software engineering are presented and suggest that using better recursive space decompositions has promise, but the additional computational effort may not be easily justified.

Keywords: Force-Directed Graph Drawing, FADE Paradigm, Recursive Space Decomposition, Quadtree, Voronoi Diagram, Recursive Voronoi Diagram, k-Means

1 Introduction

Force-directed graph drawing algorithms are a common and simple method for computing a drawing of a graph. However, they generally do not scale well as the number of nodes increases, making them unsuitable for drawing large graphs. The *FADE paradigm* addresses the problems of high computational cost and visual complexity by using *recursive space decompositions* to capture essential proximity information in the drawing [Quigley and Eades, 2000][Quigley, 2001]. This information is used to increase the speed of the force-directed graph drawing algorithm, and to create visual abstractions of the graph at various levels of detail.

In [Quigley and Eades, 2000] and [Quigley, 2001], the FADE paradigm has been presented and explored with a recursive space decomposition known as the quadtree [Samet, 1990]. Quadtrees are a commonly used recursive space decomposition which are popular as a result of their simplicity, which makes them easy to understand and implement and gives good performance. However, the geometry of a quadtree depends only on the drawing region, not the data being drawn. This means that they are not robust, and small changes to the data can result in substantially different quadtree structures. This in turn can have negative effects on the quality of the resulting graph drawing and visual abstraction.

This paper presents an exploration of the FADE paradigm using an alternative recursive space decomposition known as the *recursive voronoi diagram*. This data structure spends some extra computational effort in its construction phase in an attempt to avoid some of the problems found when using quadtrees. This provides the potential for obtaining graph drawings which are superior to those found when using quadtrees. However, preliminary results suggest that it may be difficult to justify this extra computational effort.

2 Force-Directed Graph Drawing and Recursive Space Decompositions

2.1 Force-Directed Graph Drawing

Force-directed graph drawing is a family of algorithms for drawing graphs which have become popular due to their ease of implementation and understanding, ability to handle various types of constraints, and frequent ability to draw graphs symmetrically [Di Battista et al., 1999]. They work by modelling the graph as a physical system of forces and then finding an equilibrium state of locally minimum energy of the system. Typically, edges between nodes are modelled as Hooke's law springs, and nodes are modelled as electrostatic charges. The spring forces attempt to draw adjacent nodes near one another, and the repulsive electrostatic forces prevent the resolution from deteriorating.

Since the graphs being drawn are typically sparse (that is, the number of edges m is of the order of the number of nodes n, rather than of the order of the maximum possible number of edges n^2), the computation of the long-range electrostatic forces dominates that of the springs. This is because there are $O(n^2)$ electrostatic interactions (all pairs of nodes) but usually only $O(n)$ spring interactions. This $O(n^2)$ dependance means that force-directed graph drawing methods do not scale well to large graphs.

2.2 Recursive Space Decompositions (RSDs)

Recursive space decompositions are spatial data structures which are defined recursively, and at each level of the recursion divide space into smaller regions. Each region is then further subdivided at the next level of the data structure. This creates a tree structure and allows spatial objects within these regions

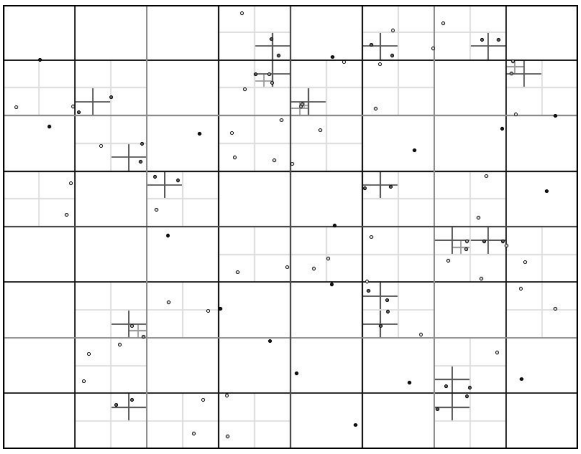


Figure 1: A quadtree spatial decomposition of 100 points uniformly distributed in the plane.

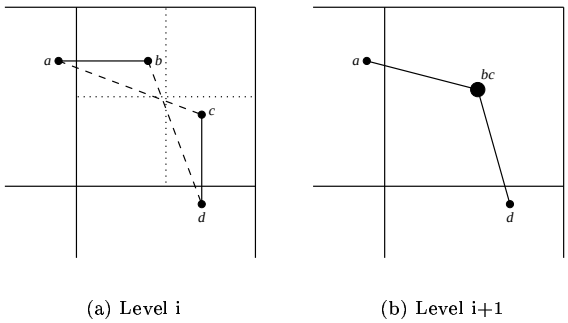
to be stored at the appropriate leaves of the tree. This is how recursive space decompositions (RSDs) store proximity information — spatial objects which are near one another in space will also be located near one another in the RSD tree. In addition to this grouping of objects, RSDs allow efficient spatial-based lookup queries. We shall restrict our attention to storing points in RSDs, and will most often use only 2 dimensions.

RSDs can be divided into two broad categories: *regular* RSDs and *irregular* RSDs. Regular RSDs divide space evenly at each level; examples include the quadtree (and variants, such as the nono-tree) [Samet, 1990] and k-d PR trie [Samet, 1990]. Irregular RSDs may divide space into arbitrarily sized and shaped regions at each level; examples include point quadtrees [Samet, 1990], k-d trees [Samet, 1990], and recursive Voronoi diagrams (Section 3). Irregular RSDs can further be categorised into *uniform* and *weighted* variants. In uniform irregular RSDs, the exact shape and size of the spatial subdivision is independant of points involved. By contrast, weighted irregular RSDs examine the data points in order to compute the spatial subdivision.

Regular RSDs suffer from not being robust, that is, the tree structure can be vastly altered by only small changes in point positions or by performing operations such as translation, rotation, and scaling on a set of points. This lack of robustness is particularly problematic for the quality of the visual abstraction (including the overall layout), as it is determined by the RSD used. Weighted irregular RSDs aim to spend some extra time at the construction stage analysing the data in order to adjust the spatial subdivisions to avoid these problems. This paper investigates using one such weighted irregular RSD, the k-means recursive Voronoi diagram, rather than a regular RSD, the quadtree, in the area of tree code optimisations for force-directed graph drawing.

Quadtrees are a commonly used regular RSD, mainly due to their simplicity, which gives them good performance without being overly complicated. They are used by the FADE paradigm analysis in [Quigley and Eades, 2000] and [Quigley, 2001]. Briefly, they work by recursively dividing space into four "quadrants" of equal size until each leaf quadrant contains at most 1 data point. They can be built in time $O(n \log n)$ for n data points and have $O(\log n)$ expected levels.

Figure 1 is an example of a quadtree spatial decomposition for 100 uniformly distributed 2D points, and Figure 2 illustrates a simple quadtree and its corresponding data structure.



(a) Level i (b) Level i+1

Figure 4: The connectivity between nodes a and d via the bc pseudonode in level $i+1$ causes the implied connectivity between nodes a and c, and nodes b and d in level i. We say that ac and bd are implied edges.

A simple example of how quadtrees can fail is shown in Figure 3. Figure 3(a) shows a set of data points well segmented by the first level of a quadtree. Common transformations which arise naturally in force-directed methods include translations, rotations and scalings. Figure 3(b) shows the result of a simple rotation of 45 degrees around the centre. The result is that now the data points are poorly segmented by the first level of the quadtree; Figure 3(c) shows a more desirable first level of the recursive space decomposition.

2.3 Tree Codes and the FADE paradigm

RSDs are used in the physics community to speed up the calculation of n-body problems; we concentrate on the treatment of Barnes and Hut [Barnes and Hut, 1986], as used in the FADE paradigm [Quigley and Eades, 2000][Quigley, 2001].

In this strategy, an RSD is constructed *ab initio* at each timestep of the force-directed graph drawing algorithm. In addition, the RSD maintains a *pseudonode* at each node of the RSD tree; this is a node located at the mean position of the descendant nodes, with weight equal to the number of descendant nodes. This is used to speed up the calculation of the node–node (ie. electrostatic forces) by eliminating the need to calculate the force contribution of every node on every other node. Instead, the force from distant nodes is included as a single group contribution by using the pseudonode. This introduces some error, but speeds up the calculation.

The exact method of computing the node–node forces is as follows. We traverse the RSD tree, and for each node in the tree, the *Barnes-Hut opening criterion*, $\frac{s}{d}$ is computed, where s is the width of the quadtree cell and d is the distance between the current node and the pseudonode of the cell. If $\frac{s}{d} \leq \theta$, where θ is a constant tolerance parameter, then the current node is deemed to be "far" and the force between the node and the cell's pseudonode is computed and used. By contrast, if $\frac{s}{d} > \theta$, traversal of the RSD continues to the children cells. The value for θ is typically near 1.0. As it approaches 0 more direct node–node interactions are computed, decreasing the error but increasing the time taken, and as it approaches infinity less direct node-node interactions are computed, increasing the error but decreasing the time taken.

Finally, there is the issue of implied edges. This is the situation where the graph at a higher level of abstraction (as defined by the RSD) can have connectivity which the lower levels do not have. An example is illustrated in Figure 4,where we say that an implied edge exists joining nodes a and c, and nodes b and d.

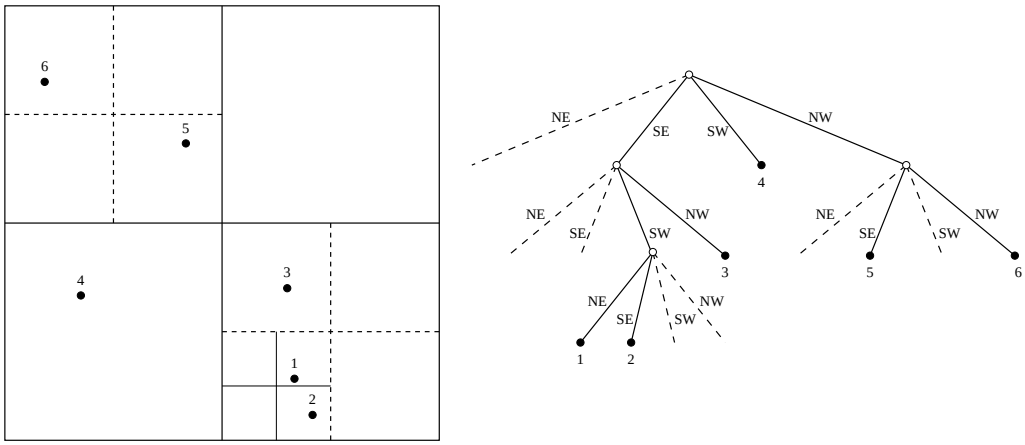
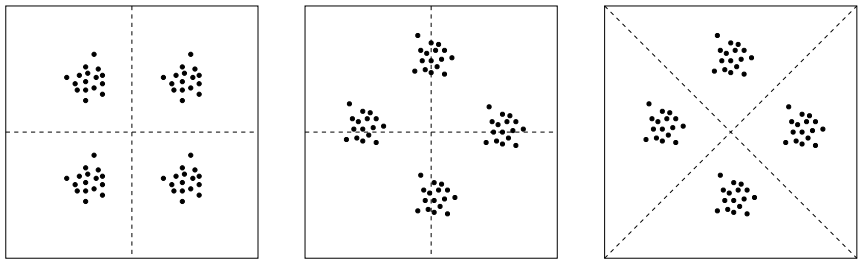


Figure 2: A simple quadtree spatial decomposition (left) and its corresponding data structure (right).



(a) (b) (c)

Figure 3: A simple illustration of how quadtrees can be less than desirable for certain point distributions.

Clearly, implied edges are not desirable where the abstracted view is to be directly viewed, and they also have an impact on the quality of the overall drawing. Indeed, the number of implied edges is a good measure of the quality of the visual abstraction. One of the aims of using a weighted irregular RSD is that implied edges caused by the non-robustness of regular RSDs can be avoided, resulting in better visual abstractions.

3 Recursive Voronoi Diagrams

This section describes the main Recursive Voronoi Diagram (RVD) data structure. It is an extension of the Voronoi Diagram (VD) to have a recursive nature.

The Voronoi Diagram is an efficient way of answering *range queries*, that is, for a set of sites in d-dimensional space (ie. $\Re^d$), which site is nearest to a given query point [Fortune, 1992][O'Rourke, 1998]. An example VD is shown in Figure 5. The VD takes as input a set S of points in space called the *sites*, and constructs as output the *Voronoi cells* corresponding to these sites. The Voronoi cell associated with site $s \in S$ is denoted by $V_S(s)$ and is the set of all points in space which are closer or equidistant to s than any other points in S. The *Voronoi Diagram* is the collection of Voronoi cells for all $s \in S$. It can be computed by various algorithms in $O(n \log n)$ time, for $n = |S|$ [Fortune, 1992].

The general idea of the RVD is to have an independant "sub-Voronoi Diagram" in every Voronoi cell. Example RVDs are illustrated in Figure 6.

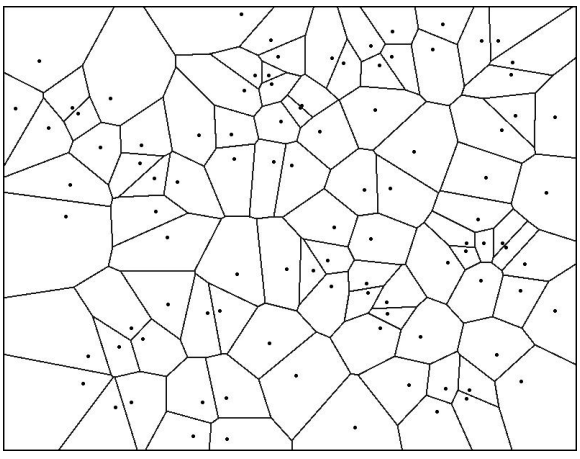
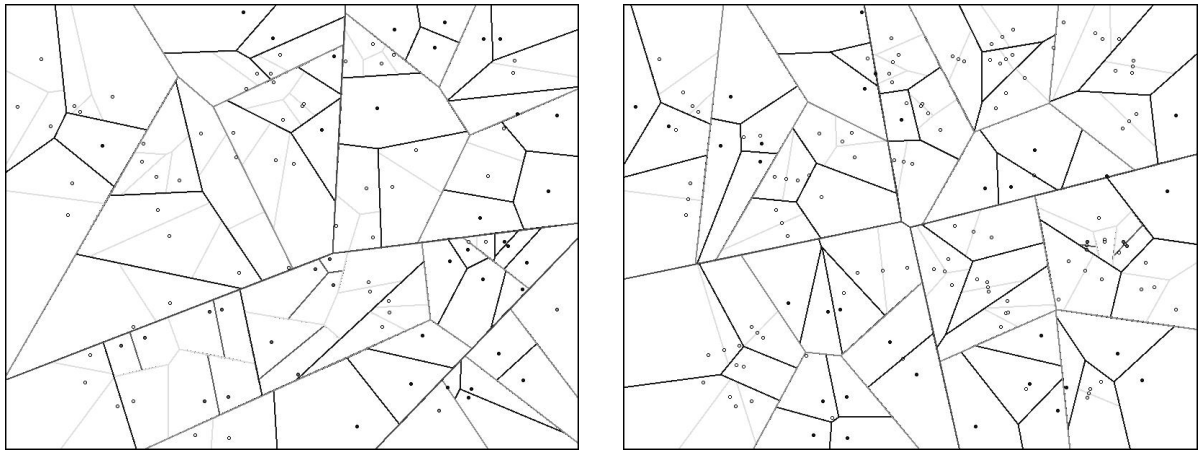


Figure 5: The Voronoi Diagram of the 100 points used in Figure 1.

3.1 Description

We define the d-dimensional RVD by presenting an algorithm for its construction. Each point will be



(a) random sampling

(b) k-means

Figure 6: Two Recursive Voronoi Diagrams of the 100 points used in Figure 1.

discussed in more detail following the outline of the algorithm.
Input:

1. A set S of d-dimensional points in d-space (ie. d-vectors), called the *sites*.
2. A connected subset $R \subseteq \Re^d$ (such that $\forall s \in S$, $s \in R$) called the *region*.

Steps:

1. Obtain a set of "characteristic" d-dimensional points C.
2. Construct the VD of the set of points C.
3. Intersect this VD with R, that is, $\forall c \in C$, $V'_C(c) = V_C(c) \cap R$.
4. Partition S into $S'(c)$, $\forall c \in C$, such that $\forall s \in S$, $s \in S'(c) \Rightarrow s \in V'_C(c)$.
5. $\forall c \in C$ recurse with $S = S'(c)$ and $R = V'_C(c)$ (unless $|S'(c)| < 2$).

The region R can be open or closed, finite or infinite. For example, it is useful to specify $R = \Re^d$ for the initial level of the RVD, or to ensure that all the Voronoi cells are are bounded by having a bounded initial R.

The method used to find the set of characteristic points C in step 1 determines the specific variant of RVD being used. This is discussed in detail in Section 3.2.

With our definition of $V_C(c)$, notice that it is possible to have $V_C(c_1) \cap V_C(c_2) \neq \emptyset$, $c_1 \neq c_2$ if c_1 and c_2 share an "edge" or "vertex" of the VD (straight lines equidistant to 2 or more sites). In the case that some $s \in S$ lies on such a Voronoi edge or vertex (ie. is in $V_C(c_1) \cap V_C(c_2) \cap \ldots$), we arbitrarily put s into exactly one of $S'(c_1)$, $S'(c_2)$, ...

We restrict our attention to $d = 2$, as higher dimensions are unnecessarily complicated for this study. In addition, the complexity of the VD scales badly as d increases: worst-case $O(n^{\lceil \frac{d}{2} \rceil})$ (for $n = |S|$) and expected-case $O(c_d n)$ where c_d is a constant which grows exponentially with d, making VDs in higher dimensions impractical [Fortune, 1992].

3.2 Finding the characteristic points

Various strategies are possible for obtaining suitable sets of characteristic points C. In each case, the number of characteristic points may or may not be constant. For example, it may depend on the number of sites n, or the level of the RVD l.

Uniform Distribution. This method chooses characteristic points according to a uniform distribution (with no respect for the data points). This requires the initial R to be bounded (and hence every Voronoi cell also bounded), so that the characteristic points can be chosen uniformly in these regions. It takes time $O(r)$, where r is the number of data points chosen.

Random Sampling. This method chooses characteristic points by randomly selecting r sites. It also takes time $O(r)$.

k-means. This method runs the k-means clustering algorithm on the sites and then chooses the characteristic points to be the k representative points of the k-means clustering. The k-means algorithm can either be run to convergence (which isn't guaranteed) or for a constant number of iterations. The k-means algorithm takes time $O(kn)$ per iteration.

The k-means algorithm is a method of geometric clustering [Estivill-Castro and Houle, 2001]. This means that in a graph drawing context, the clustering of the nodes is obtained by considering only the locations of the nodes, without regard for any edges.

Figure 6 illustrates two example RVDs on the same set of points. Figure 6(a) uses the random sampling method of choosing characteristic points; notice that the Voronoi cells have poor aspect ratios and can be quite irregularly shaped. By contrast, Figure 6(b) uses the k-means method of choosing characteristic points; notice that the Voronoi cells have better aspect ratios, the divisions are more regular and distributed, and the depth of the RVD tree is in general smaller.

The k-means algorithm works by storing a *representative point* for each cluster of points. This representative is the mean of the points in the cluster. Each point is then assigned to the cluster corresponding to its nearest representative point. The representative points are then recalculated based on the new cluster points, and the process iterates. The initial clustering for k-means is a random clustering, however the clustering is not reset between iterations of the FADE

algorithm. This prevents the k-means algorithm from needing to escape from the initial random clustering at every FADE iteration, since the clustering carried over from the preceeding FADE iteration is a good starting point for the current iteration's clustering.

It is possible to use geometric clustering methods which are more advanced than k-means, such as the k-medoids based methods presented in [Estivill-Castro and Houle, 2001], although they are generally more expensive.

3.3 Complexity analysis

If we consider the number of characteristic points $|C| = k$ to be constant at each level, then we would expect that each cell will have a constant fraction of the number of points, for a uniform distribution of original points. This implies that we expect $O(\log n)$ levels (for $n = |S|$) for an approximately uniform distribution of characteristic points. Finding the Voronoi Diagram of the characteristic points takes time $O(k \log k)$, which is $O(1)$ (since k is constant). It also takes constant time to find the cell a query point lies in.

During construction, each point will be partitioned into a single cell at each level, taking $O(n \log n)$ time. If the k-means method of finding characteristic points is used, then at each level the cost due to this will be $O(n)$ (for a constant number of k-means iterations). This gives a total k-means cost of $O(n \log n)$, and a corresponding total overall construction cost of $O(n \log n)$. This is the same as the construction cost of a quadtree, however the hidden constants involved with the RVD are generally worse (due to the overheads involved in computing the Voronoi Diagram and the k-means algorithm).

If $|C|$ is not constant, but rather $|C| = c\,|S|$ for some constant fraction $0 < c \leq 1$, then the k-means cost per level becomes $O(kn) = O(n^2)$ (since now $|C| = k = O(n)$). The cost of taking the Voronoi Diagram is $O(n \log n)$ time, and it takes $O(\log n)$ time to find the cell a query point lies in. However, we only expect $\frac{n}{k} = \frac{n}{cn} = \frac{1}{c}$ points per cell on the second level, which gives a constant number of levels. The overall construction time in this case is dominated by the $O(n^2)$ to run k-means, which is as poor as the original force-directed graph drawing algorithm. For this reason, we restrict our attention to the case where $|C|$ is constant. Choosing $|C| = c\,|S|$ may give benefits despite the increased computational complexity, but this is outside the scope of this paper.

3.4 Cell size measures

Recall from Section 2.3 that the Barnes-Hut opening criterion θ is a measure of the distance of the current node from the centroid of the query cell, with respect to the size of the query cell. In particular, a cell is opened when $\theta > \frac{s}{d}$ for s the query cell's size, d the distance between the current node and the query cell's pseudonode (center of mass), and θ the constant tolerance opening criterion.

For quadtrees, choosing s to be the width of the query cell is natural and straightforward. However, for irregular RSDs such as the Recursive Voronoi Diagram, the "size" s of each cell may not be so obvious.

The measure used is to take the average of the width and height of the smallest enclosing rectangle aligned with the x and y axes (said to be an "orthogonally-aligned rectangle"). This was chosen because it is a simple measure which seems to work well in practice. Alternatives include:

1. the square root of the area of the smallest enclosing orthogonally-aligned rectangle,

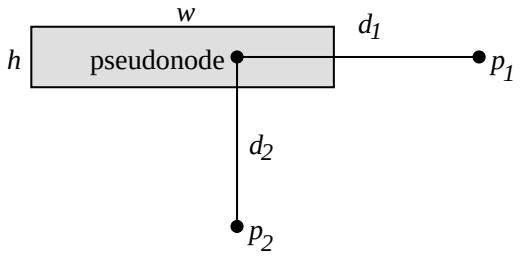


Figure 7: An example cell suggesting the use of moments.

2. the square root of the area of the smallest enclosing circle,
3. the radius of the smallest enclosing circle, and
4. the square root of the area of the cell.

All of these involve more computation than our chosen measure.

Figure 7 illustrates an example where taking the average of the width and height or the square root of the area of a rectangular cell both give a less than desirable result. Consider two points p_1 and p_2, where p_1 is distance d_1 from the cell's pseudonode and p_2 is distance d_2 from the cell's pseudonode. If we make d_2 marginally smaller than d_1, it is possible to choose w and h such that using $s = \sqrt{wh}$ gives $\frac{s}{d_1} > 1$ (ie. p_1 is far) and $\frac{s}{d_2} < 1$ (ie. p_2 is close). However, p_1 is closer to the cell as a fraction of the cell width w, than the distance of p_2 as a fraction of the cell height h. This suggests a method which takes the moment of the cell to obtain the smallest enclosing rectangle (in any orientation). This rectangle could then be used as outlined above for the orthogonally-aligned rectangles. However a more sophisticated method could find the distance to the cell in terms of the axes of this rectangle. This would more accurately represent the distance between the cell and point in terms of the shape and orientation of the cell.

4 Preliminary Results

This section presents some preliminary results obtained using the k-means RVD with the FADE paradigm of force-directed graph drawing. More extensive results are currently being gathered and analysed.

We examine two sets of graphs. The first are random "node clustered" graphs, and the second are a small selection of graphs in the domain of software engineering from the Bauhaus project [Bauhaus, 2001], as used and described in detail in [Quigley, 2001]. The results are presented in the form of a picture gallery (Figures 8 to 12) comparing the direct force-directed method (ie. without FADE), the quadtree based FADE method, and the k-means RVD based FADE method. More complete results including graph based measures (such as the number of edge crossings and the aspect ratio), clustering measures (such as the Cohesion and Coupling (COCO), Implied Edge Precision (IEP), Lowest Common Ancestor (LCA) and Node Neighbourhood Similarity (NNS)), and timing and error rates will be included in the aforementioned extensive results.

Random "node clustered" graphs are generated in the following way. The number of nodes n and edges m are specified as input; typically $m = cn$, where c is around 1.05. The number of desired clusters k is also specified as input. Of the n nodes, k are randomly selected to be the "cluster nodes". Each of the m edges is created by choosing a source and destination node

such that each edge is equally likely to be a cluster node or not. Thus, half of the time, the source node is chosen to be one of the k cluster nodes, the remaining half of the time, the source node is chosen to be one of the $n - k$ non-cluster nodes. The destination nodes are chosen in the same way. This means that it is equally likely for an edge to be between two cluster nodes or two non-cluster nodes, and it is twice as likely for an edge to be between a cluster node and a non-cluster node.

This experiment used $k = 4$ for both the random node clustered graph generation and in the k-means clustering algorithm (ie. $|C| = 4$), so as to be consistent with the number of subdivisions made by the quadtree FADE method. Most of the graphs were not connected, so to avoid problems with different connected components interfering with one another, only the largest connected component was used. The graph drawings are informally and subjectively compared to determine how relatively "crowded" they are in terms of areas of high node or edge density. This is based on the reasoning that a more robust RSD has less implied edges, thereby giving a better approximation of the direct node-node forces. More accurate node-node forces should be able to better overcome the edge forces, giving a drawing with more evenly spaced nodes.

As can be seen in Figures 8 and 9, the difference between the quadtree and k-means RVD FADE methods is small for random node clustered graphs. For Figure 8, the difference between quadtree and RVD is difficult to see, and for Figure 9, differences are noticable but it is unclear which is better.

The drawing produced by quadtree in Figure 10 is similar, but slightly worse than that produced by direct and RVD. There is a small group of nodes on the right side of the main central arrangement of nodes which is obscured in the quadtree layout. The entire right side is clearest in the RVD layout, followed closely by the direct layout.

There are no substantial differences between any of direct, quadtree and RVD in Figure 11.

The drawing produced by RVD in Figure 12 is noticeably worse than those produced by the direct and quadtree methods, as the RVD layout is much more congested.

5 Conclusions and Future Work

Recursive space decompositions (RSDs) can be used to improve the time performance of force-directed graph drawing algorithms. Current investigations of the FADE paradigm have concentrated on regular RSDs, in particular the quadtree. We have presented an alternative FADE algorithm, based on the a weighted irregular RSD called the k-means recursive Voronoi diagram and compared it to the quadtree based FADE algorithm.

The preliminary results presented in Section 4 tend to suggest that the layouts produced by the k-means recursive Voronoi diagram FADE algorithm are generally of comparable quality to those produced by the quadtree FADE algorithm. In some cases, the k-means RVD FADE algorithm performs better, but in other cases the quadtree FADE algorithm does. Complete results, including objective measures such as graph based measures and clustering measures, an examination of the visual abstraction (in particular the implied edges) and results for more varied and larger graphs, are needed to fully resolve the issue of what class of graphs or circumstances may benefit from the use of the k-means RVD FADE algorithm.

Future work can investigate the possibility of using other weighted irregular RSDs which are cheaper than the RVD but still perform better than quadtrees. Candidates for this work are the weighted k-d PR trie [Samet, 1990] and the 2-RVD (the k-means RVD with $k = 2$, similar to the weighted k-d PR trie). Also interesting is the possibility of uniform irregular RSDs, or even some restricted class of regular RSDs, which avoid the robustness problem. Finally, an investigation of the effect of cell size measure (Section 3.4) on the performance of irregular RSDs could be carried out.

References

[Barnes and Hut, 1986] Barnes, J. and Hut, P., (1986). A hierarchical $O(n \log n)$ force-calculation algorithm, *Nature*, **324**, no. 4, 446–449.

[Bauhaus, 2001] Bauhaus project web site. http://www.informatik.uni-stuttgart.de/ifi/ps/bauhaus/index-english.html

[Di Battista et al., 1999] Di Battista, G., Eades, P., Tamassia, R. and Tollis, I. G. (1999). *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall, Upper Saddle River, New Jersey.

[Estivill-Castro and Houle, 2001] Estivill-Castro, V. and Houle, M. E. (2001). Robust Distance-Based Clustering with Applications to Spatial Data Mining, *Algorithmica*, **30**, 216–242, Springer-Verlag.

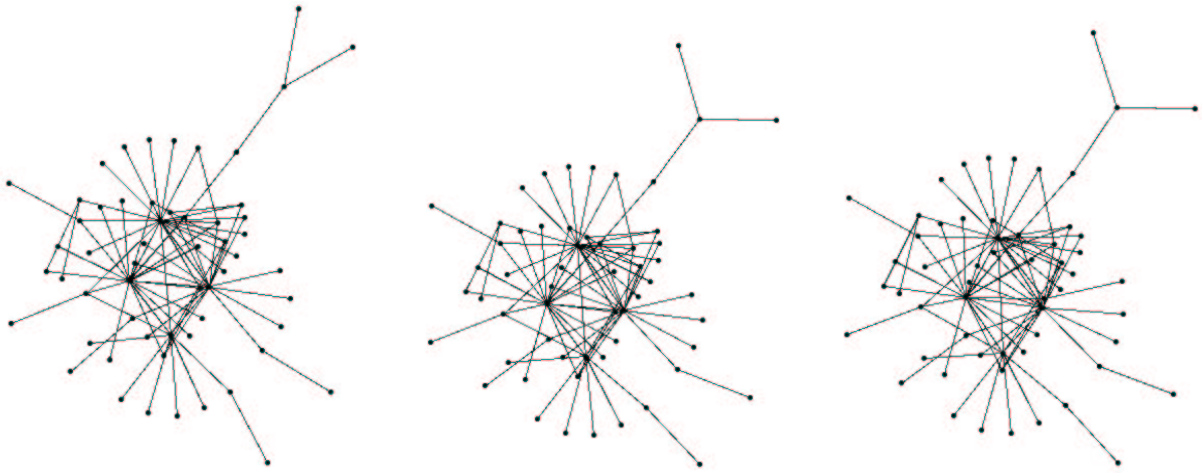
[Fortune, 1992] Fortune, S. (1992). Voronoi Diagrams and Delaunay Triangulations, *Computing in Euclidean Geometry*, Ding-Zh Du and Frank Hwang (eds), World Scientific, Singapore.

[O'Rourke, 1998] O'Rourke, J. (1998). *Computational Geometry in C, Second Edition.* Cambridge University Press, New York.

[Quigley, 2001] Quigley, A. (2001). *Large Scale Relational Information Visualization, Clustering, and Abstraction.* PhD thesis, University of Newcastle, Australia.

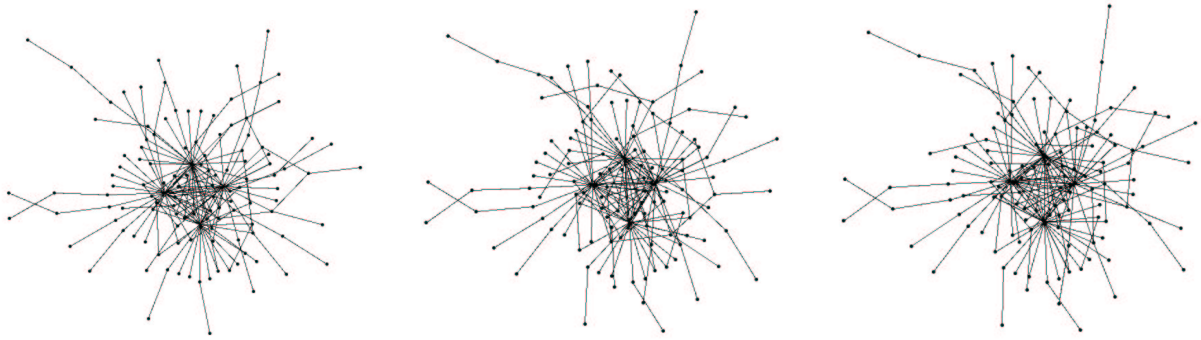
[Quigley and Eades, 2000] Quigley, A. and Eades, P. (2000). FADE: Graph Drawing, Clustering, and Visual Abstraction, *Proceedings of Graph Drawing 2000*, Joe Marks (ed), Springer LNCS 1984.

[Samet, 1990] Samet, H. (1990). *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, Reading, Massachusetts.



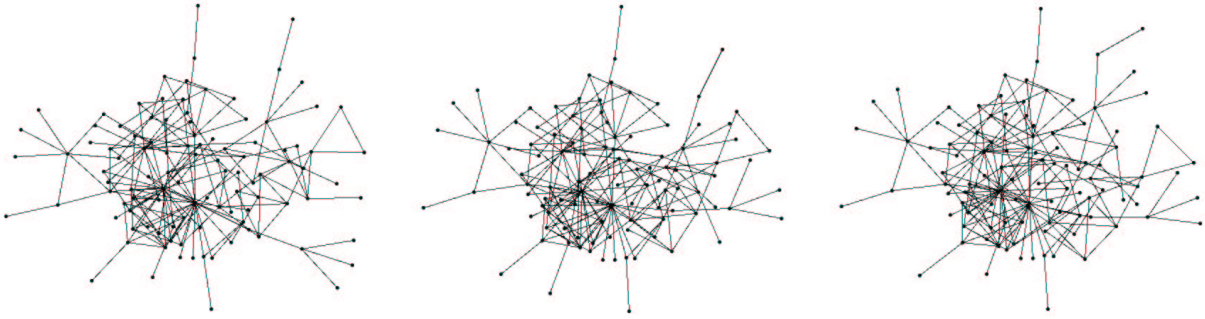
(a) direct (b) quadtree (c) k-means RVD

Figure 8: Random node clustered graph with 57 nodes and 99 edges.



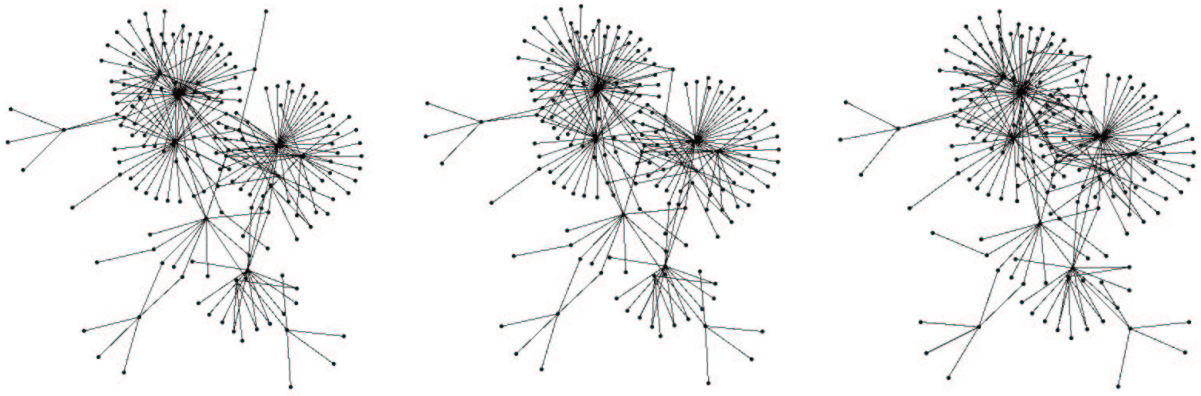
(a) direct (b) quadtree (c) k-means RVD

Figure 9: Random node clustered graph with 123 nodes and 200 edges.



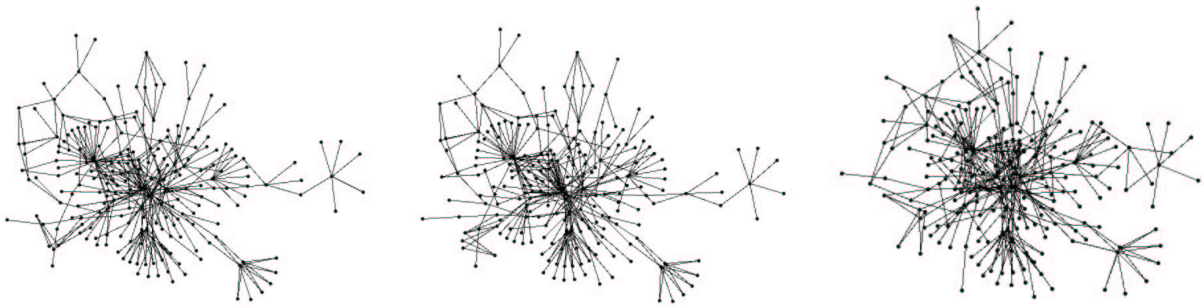
(a) direct (b) quadtree (c) k-means RVD

Figure 10: The largest connected component of the *same expression view* graph for the "bash" software system, from the Bauhaus project. Contains 104 nodes and 174 edges.



(a) direct (b) quadtree (c) k-means RVD

Figure 11: The largest connected component of the *signature view* graph for the "bash" software system, from the Bauhaus project. Contains 177 nodes and 211 edges.



(a) direct (b) quadtree (c) k-means RVD

Figure 12: The largest connected component of the *actual parameter view* graph for the "bash" software system, from the Bauhaus project. Contains 220 nodes and 296 edges.