# Mining Scale-free Networks using Geodesic Clustering *

Andrew Y. Wu
Dept. of Computer Science
University of Illinois
Urbana-Champaign, IL 61801
a.wu@acm.org

Michael Garland
Dept. of Computer Science
University of Illinois
Urbana-Champaign, IL 61801
garland@uiuc.edu

Jiawei Han
Dept. of Computer Science
University of Illinois
Urbana-Champaign, IL 61801
hanj@cs.uiuc.edu

## ABSTRACT

Many real-world graphs have been shown to be scale-free—vertex degrees follow power law distributions, vertices tend to cluster, and the average length of all shortest paths is small. We present a new model for understanding scale-free networks based on multilevel geodesic approximation, using a new data structure called a *multilevel mesh*.

Using this multilevel framework, we propose a new kind of graph clustering for data reduction of very large graph systems such as social, biological, or electronic networks. Finally, we apply our algorithms to real-world social networks and protein interaction graphs to show that they can reveal knowledge embedded in underlying graph structures. We also demonstrate how our data structures can be used to quickly answer approximate distance and shortest path queries on scale-free networks.

## Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—Data Mining

## Keywords

graphs, social networks, scale-free networks, clustering

## 1. INTRODUCTION

In recent years, the analysis of structurally rich graph data sets has received increasing amounts of attention in data mining and related disciplines. The core problem of graph mining [17] arises in many important problem domains. For example, recent work in the related area of mining social networks includes the study of viral marketing [4, 15] and of measuring the relative importance of nodes [19]. Graph and social network mining have also been used to find hubs in hyperlinked corpora [3, 9] and to detect community structure in social networks [13].

Our focus is on the class of graphs termed *scale-free networks*. Graphs of this type are distinguished by three primary characteristics. First, they are highly clustered; if two vertices share a common neighbor, it is likely the two are themselves adjacent. Second, the average shortest path between two vertices is logarithmically small. And finally, the vertex degrees are distributed according to a power law [1]. Data fitting this profile arise quite naturally in physics, sociology, network analysis, and biology.

In this paper we present a scalable framework for analyzing the structure of scale-free networks. We approach this problem from the perspective of data reduction. Given an initial complex graph, we aim to produce a far simpler graph that preserves the structure of the original as faithfully as possible. We describe a novel algorithm for clustering graphs based on graph geodesics (i.e., shortest paths). Furthermore, we outline a hierarchical model for scale-free networks, and couple this with our clustering algorithm to produce hierarchical representations of the input data.

## 2. BACKGROUND

We use the terms graph, network, and mesh interchangeably to mean an undirected simple graph with positive edge weights. We define a mesh clustering of a graph $G = (V, E)$ as a graph partition into $p$ disjoint clusters $V_1, \ldots, V_p$ where $\bigcup_i V_i = V$, with a representative vertex set $U = \{u_1, \ldots, u_p\}$ where $u_i \in V_i$. A median function $M$ maps vertices to their representative vertices (medians). That is, $M(v) = u_i$ iff $v \in V_i$. A graph partition of a graph $G$ is a vertex clustering where each vertex cluster forms a connected component.

Many real-world systems can be modeled as graphs. Examples include power grids, communication networks, biochemical interactions, and social networks. On social networking sites such as orkut.com and Friendster, an edge between Paul and Carol exists if they specify that they know each other. In a publication network, edges connect coauthors. Although such graphs capture many aspects of the real world, they also capture the complexity of the real world. Thus, it becomes important to develop mathematical models for how very large graphs form. We can then use these models to mine knowledge from massive graphs.

In the Erdös-Rényi random graph model, edges are added by picking random vertex pairs and the degree distribution is Gaussian [2]. Several years ago, it was discovered that many

*This work was supported in part by the National Science Foundation NSF CCR-0098170, NSF IIS-02-09199, and the University of Illinois at Urbana-Champaign. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.


*KDD'04,* August 22–25, 2004, Seattle, Washington, USA.

real-world networks do not follow this random model [1]. Instead, they often exhibit power law distributions, where the probability that a vertex has degree $k$, $P(k) \sim 1/k^c$ for some small constant $c$ (often between two and three) [12].

This means that most vertices have a relatively small degree, but a few vertices have a very high degree. These are known as hubs, and are well connected. Unlike in a Gaussian distribution, where a mean element defines a characteristic "scale" in the probability distribution, no single element characterizes the scale of a power law distribution. Thus, Barabási named such graphs scale-free networks [1].

To perform data mining on such graphs, we exploit their scale-free nature, noting that the power law distribution tells us that most vertices are of low degree. This means that the average degree of a vertex is generally bounded by a small constant. Since the sum of the vertex degrees must be twice the number of edges, the total number of edges in the graph $G$ is linear in the number of vertices ($G$ is sparse). We also exploit the tendency of these networks to cluster to perform good data reduction on such graphs.

Barabási showed that the world-wide web, for example, is a scale-free network [1]. Scale-free characteristics have been found in many real-world networks, such as social networks, publication and citation graphs, power grids, the Internet, and real neural and other biological networks. More details can be found in the recent survey of Newman [12].

## 2.1 Social Networks

Sociologists have defined many ways of measuring the centrality or prestige of an "actor" in a social network [18]. One such measure, betweenness centrality, has been used to find community structures in networks [13]. However, a simple implementation of these measures often leads to quadratic or cubic running times, because they often compute all-pair shortest paths, or invert matrices. In contrast, we focus on scalable algorithms for very large graphs.

## 2.2 Data Mining

Most work on clustering considers unorganized points instead of considering the network structure of graph systems, as is common in graph partitioning [16]. Graph mining research [17] has studied problems of finding frequent subgraphs [20] and compressing graphs using the minimum-description length principle [7].

In comparison, we focus primarily on the problem of graph clustering and data reduction of large real-world graphs. One problem of working with graphs as opposed to unorganized data points is that most non-trivial graph properties are computationally difficult to verify, and many simple graph algorithms have exponential running times. Just as microclustering has been useful for speeding up algorithms that run on unorganized point sets [21], we suggest that graph clustering and data reduction techniques can be used to speed up data mining of very large graph data.

# 3. APPROXIMATION CLUSTERING

Given a complex graph representing some data set, we would like to extract meaningful knowledge from that graph. However, if the graph has thousands or millions of vertices, it becomes difficult to visualize or mine that graph algorithmically in any meaningful or efficient way.

A common approach to dealing with such complexity is to find meaningful clusters or partitions in the graph. However, there are dozens if not hundreds of different clustering algorithms. As suggested by Mirkin [11], we evaluate the quality of a clustering in terms of the approximation error induced by that clustering.

Given a clustering of a graph $G$, we can approximate $G$ with a smaller graph $G'$ by contracting each cluster to a representative vertex. To quantitatively measure how much the contraction of a clustering distorts a graph, we can measure how much it distorts the lengths of all possible geodesics.

In this section, we describe how to find geodesic clusters on graphs, how to contract these clusters to form an approximate graph, and how to measure the distortion (approximation error) associated with a cluster contraction.

## 3.1 Geodesic Clustering

We now describe a novel graph clustering technique designed for large sparse graph data. Later, we will use this clustering technique as a basis for graph data reduction.

Given a weighted graph, we want to find $p$ representative vertices, which we call median vertices, that minimize the average length of the shortest path from any vertex to its closest median. In operations research, this is known as the network p-median problem in discrete facility location [5].

Formally, we define $M(v)$ as the median vertex associated with $v$'s cluster and $\text{dist}(x, y)$ to be the length of the shortest path from $x$ to $y$. We also define $D(v)$ to be $\text{dist}(v, M(v))$, the distance from a vertex to its nearest median vertex. Then we want to find the mesh clustering with mapping $M$ that minimizes the contraction cost $\sum_{v \in V} D(v)$. We describe how to pick good medians in a future section.

After finding $p$ medians, we assign each vertex to its nearest median vertex, forming $p$ geodesic clusters. The clustering algorithm keeps track of two properties per vertex $v$: the nearest median vertex, $M(v)$, and the shortest path distance to that nearest median, $D(v)$. Initially, distances are infinite and each vertex is in a singleton set. We chose $p$ vertex medians and assign each a distance of zero. We can then run a modified version of Dijkstra's algorithm that updates the "parent" of a vertex (its closest median) whenever the vertex's distance is updated. In the resulting graph, each vertex has a single distance to its nearest median and a "parent" pointer to that same median vertex. This algorithm runs in $O(V \log V)$ time on a sparse graph system.

## 3.2 Graph Approximation

Our approach for data reduction on large social and scale-free networks is inspired by work on geometric mesh simplification in the graphics community [6]. Given a graph $G$, we want to find a simpler graph $G'$ that well approximates $G$ (minimizing some distortion measure).

To solve this primal problem of mesh simplification, we consider the dual problem of mesh clustering. Given a mesh clustering of a graph $G$, we construct a simpler graph $G'$ by contracting each cluster to a representative vertex.

We give the pseudocode for cluster contraction as Algorithm 1. The cluster contraction algorithm creates a new approximate graph $G'$ given a clustered graph $G$. It first copies the representative vertices, those for which $M(v) = v$. $(u, v)$ is an edge of $G'$ if and only if there exists an edge $(s, t)$ in $G$ such that $M(s) = u$ and $M(t) = v$. We define the new edge weight as the length of the shortest path between $u$ and $v$ in $G$ that has an edge with endpoints in $u$'s cluster and $v$'s cluster.

**Algorithm 1:** Contract Clustering

**Input**: Clustered graph G = (V, E)
**Output**: Approximate graph G'

```
for each vertex v in V do
    if v = M(v) then copy v to G'
end
for every edge e = (u, v) in E do
    if M(u) ≠ M(v) then
        d ← D(u) + length(e) + D(v);
        Create edge e' = {M(u), M(v)} in G' if not present;
        length(e') ← min(length(e'), d);
    end
end
```

# 4. SCALE-FREE NETWORKS

We now apply the geodesic clustering techniques we have developed to scale-free networks such as social networks and protein-protein interaction graphs.

## 4.1 Multilevel Mesh

We propose a new way of understanding and mining scale-free networks, using a new data structure called a multilevel mesh. A multilevel mesh is a hierarchy of microclusters similar to the tree structure used in BIRCH [21], but designed specifically for graph clustering.

Unlike a regular cluster hierarchy, a multilevel mesh is not a tree. It is a list of graphs $L = (G_0, G_1, \ldots, G_n)$ where $G_{i+1}$ is a simplified version of $G_i$, with virtual edges that map every vertex from $G_i$ to its parent vertex in $G_{i+1}$.

We can construct a multilevel mesh by recursive graph simplification—geodesic clustering followed by cluster contraction. We begin with a graph $G$ and produce a list of graph approximations of decreasing complexity. The user chooses an average branching factor $b$, which determines the average cluster size. If $b = 10$, then the average cluster will contain ten vertices. Dividing the number of vertices $|V|$ by $b$ gives us $p$, the number of clusters we need to find. We find $p$ geodesic clusters as described previously, using some heuristic to find median vertices. We contract each cluster to its representative vertex to form a new graph $G'$ with $p = |V|/b$ vertices, and recurse on $G'$.

## 4.2 Modeling Scale-free Networks

Vertices in scale-free networks are not created equal—some are more "important" than others. A few Web pages are linked to more frequently and thus may be more relevant for keyword search [9, 3]. Some actors in a social network are better connected than their peers, and some proteins are more important to an organism's survival than others [8].

Exploiting these power law distributions and the tendency for such networks to cluster, we use the multilevel techniques we have developed so far to mine such networks. The unequal distribution of importance and the high connectivity of hubs means that we can approximate a graph well using only a relatively small fraction of the vertices. The tendency for vertices to cluster reduces the amount of approximation error when building such graph approximations.

As an example, we could build a multilevel mesh that approximates a social network, favoring well-known people. Instead of a large population of social actors, we can instead imagine a smaller set of community leaders as well as the interaction graph between those leaders.

Now that we have defined a multilevel model for understanding scale-free networks, we show how to mine such networks to extract a multilevel structure. We use this self-similar hierarchy of graphs to perform data reduction, visualization, and approximate distance queries on large graphs.

## 4.3 Clustering Scale-free Networks

Previously we showed how to find geodesic clusters on graph systems, but did not describe how to find representative vertices. We experimented with several methods for finding $p$ median vertices on scale-free networks: random sampling, degree ranking, HITS, and Betweenness-Centrality.

In random sampling, we pick $p$ median vertices uniformly at random, without replacement. In degree ranking, we pick the $p$ vertices with the highest degree. We also used HITS [9] to find good median vertices, sorting vertices by their "authority" measure. Finally, we used an importance measure named Betweenness-Centrality previously used for finding community structures [13]. Betweenness-Centrality estimates the importance of a node by counting how many shortest paths pass through that node. Each of these methods define a total ordering of the vertex set, and to pick $p$ medians we choose the $p$ highest ranked vertices.

## 4.4 Approximate Distance Queries

A fundamental query operation on a graph is to return the length of the shortest path between two vertices, or to return the shortest path itself. Unlike the computation of distance queries between points in Euclidean space, we cannot answer a distance query on a large graph in constant time unless we store all possible answers. With massive graph data, this may not be feasible since there are a quadratic number of vertex pairs. On the other hand, a simple single source shortest path such as Dijkstra's algorithm may explore many parts of a graph, especially in scale-free networks where the average path length is logarithmically small.

We can use our graph hierarchy to quickly approximate shortest path queries, without touching as much of the graph as a standard search (Algorithm 2). It only uses two graphs, but is easy to extend to use more levels, depending on the accuracy desired. We call this a focused search method because we use the graph hierarchy to focus our geodesic search on a relatively small part of the graph.

**Algorithm 2:** Find Approximate Shortest Path

**Input**: Base graph $G_0$, Simpler graph $G_1$, Vertex source, Vertex target
**Output**: Path P

Find shortest path P' from $M$(source) to $M$(target) on $G_1$;
Find shortest path P between source and target on $G_0$, not exploring vertex $v$ if $M(v)$ is not on path P'

By varying the branching factor and number of clusters, we can tradeoff between efficiency and approximation error. These distance queries can be used as a building block to speed up more complicated graph analysis techniques.

## 4.5 Experiments

We compare several heuristics for picking median vertices on scale-free network data, and demonstrate how we can trade-off between graph approximation error and distance query times. We show that random sampling is useful for approximating distance (path length) queries efficiently, but

| data set | # vertices | # edges |
|---|---|---|
| smyth.net | 286 | 554 |
| c-erdos971.net | 429 | 1312 |
| c-erdos972.net | 5440 | 14382 |
| hep-th.net | 5835 | 19652 |
| protein-bo.net | 1458 | 1948 |

**Table 1: Graph data sets**

| data set (# hubs) | B-C | Degree | HITS | Random |
|---|---|---|---|---|
| c-erdos971 (12) | 1.7 | 1.9 | 1.9 | 2.8 |
| c-erdos972 (100) | 1.5 | 1.5 | 1.9 | 2.5 |
| hep-th (287) | 1.8 | 2.0 | 2.9 | 2.3 |

**Table 2: Average distance to nearest hub**

does not work as well for approximating the paths themselves, or for yielding informatively simplified graphs. We show that Betweenness-Centrality is a good heuristic for choosing median vertices but that ranking by vertex degree is often nearly as good, and can be used as a simple and efficient heuristic for finding geodesic clusters to contract.

We implemented our algorithms in Java using the JUNG framework [14], which provides a graph library and visualization capabilities. We used several data sets, mainly publication networks in addition to a protein interaction network. In each data set, we considered only the largest connected component which contains the majority of the vertices.

Smyth.net is a publication network centered around Dr. Padhraic Smyth. We use two subsets of the Erdös publication network, and a publication network of theoretical high energy physicists (hep-th). Protein-bo.net is a scale-free protein-protein interaction graph, where each vertex is a protein and edges connect interacting proteins. This data set was studied by Jeong et al. who found that the degree of a protein in this graph correlates to its lethality [8].

In these experiments, we used unit edge weights. If the graph data provided edge weights, or if there were a reliable way of finding appropriate weights given some application domain, our algorithms can also use that information.

### 4.5.1 Average hub distance

When choosing hubs (representative vertices), we would like to pick hubs that are quickly reachable from any other vertex. Table 2 contains the average distances to the nearest hub for several data sets. We see that choosing hubs based on Betweenness-Centrality (B-C) and on vertex degree minimizes the average distance to the nearest hub. Random sampling and HITS often do significantly worse.

If our goal is to uniformly spread the graph distortion across all vertices, then we would like to minimize this average. Such a measure is also a fast way of approximating the potential influence of a set of nodes in a social network. That is, if we pick an "influential" set of people in a social network, we can expect that they can more quickly spread information throughout the network. These experiments support the commonsense notion that to quickly reach many people in a network, a simple heuristic that works well is to target the most "popular" people, in terms of vertex degree.

### 4.5.2 Distance Approximation Error

Table 3 shows the average percent difference between the lengths of paths on simplified graphs as compared to actual

| data set (b) | B-C | Degree | HITS | Random |
|---|---|---|---|---|
| c-erdos972 (10) | 20% | 20% | 34% | 18% |
| hep-th (10) | 24% | 24% | 41% | 17% |
| protein-bo (6) | 18% | 16% | 26% | 13% |

**Table 3: Avg. Distance Approximation Error**

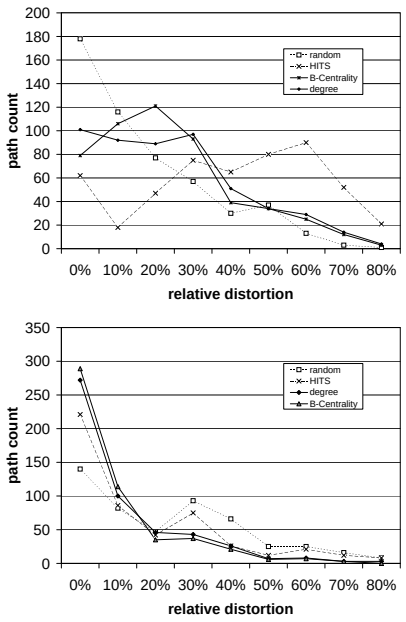


**Figure 1: Distortion, w/o focused search (above) and with focused search (below), hep-th, b = 10**

lengths, measured over a random sample of 512 vertex pairs. On the c-erdos972 data, we see that random sampling, B-Centrality, and degree ranking all result in a similar amount of distortion, about 20%, which is relatively small when we consider we are reducing the size of the graph by an order of magnitude. Overall, random sampling often results in a lower overall distortion. We believe that this occurs because the vertices are chosen without bias, whereas degree ranking and HITS may spread the distortion unevenly.

### 4.5.3 Path Approximation Error

In the previous section, we measured the relative error involved in approximating distance queries with a single-level graph approximation. We now measure the same relative error, but test the algorithm that finds better approximate geodesics using a two-level focused search (Algorithm 2).

Comparing the two histograms in Figure 1, we see that a two-level focused search obviously leads to a smaller path approximation error (about half) when compared to a single-level approximation.

Table 4 shows the average relative geodesic approximation error when finding the shortest path between 512 randomly chosen vertex pairs. Looking at this table and at Figure 1, we can see that Betweenness-Centrality leads to the smallest approximation error, but that degree ranking does nearly as well. But, because of the $O(V^2)$ cost of computing Betweenness-Centrality, we generally suggest the use of degree ranking for purposes of geodesic approximation.

| data set (b) | Degree | HITS | Random | B-C |
|---|---|---|---|---|
| c-erdos972 (64) | 12% | 9% | 17% | 12% |
| hep-th (20) | 18% | 26% | 22% | 13% |
| hep-th (10) | 12% | 18% | 25% | 9% |
| protein-bo (6) | 4% | 13% | 16% | 5% |

**Table 4: Avg. Path Approximation Error**

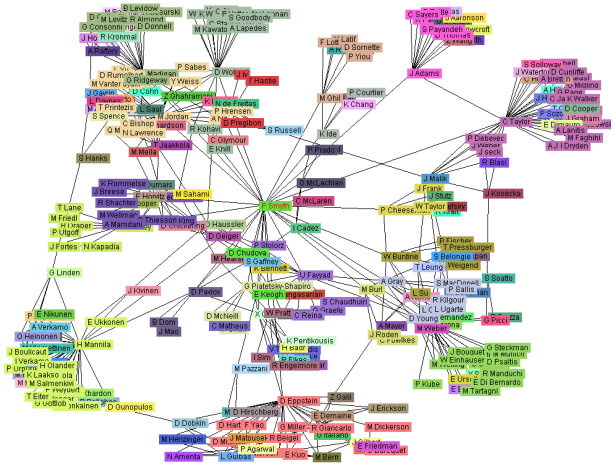

**Figure 2: Smyth.net – 286 vertices**

## 4.6 Simplified graph visualization

We can also use graph approximation to visualize very large graphs. When graphs of thousands of vertices are displayed, a user cannot easily make sense of the data. However, if we reduce the data size while still retaining information about the graph topology, the user can visually understand the underlying graph patterns in the data.

They can also perform a kind of roll-up and drill-down by shrinking clusters to their representative vertices, or expanding a representative vertex to its original cluster.

As a concrete example, a manager may want to understand the structure of her organization based on how employees interact, to see which divisions tend to collaborate. An electrical engineer might want to roll-up a power grid to understand how the overall system is operating.

We give several visual examples of how graph simplification can reveal underlying patterns in graph connectivity. (Simplified graphs are laid out independently of the base graphs, which is why the geometric layouts do not match. Similarly shaded vertices are in the same clusters.)

Figures 2 and 3 show the Smyth publication network simplified using degree ranking, which favors well-known social actors. We see that the resulting graph of 17 vertices contains well-known researchers and makes it easy to see how researchers from different subfields connect to each other.

The protein interaction graphs in Figure 4 were both laid out automatically using a simple spring layout. Whereas this simple algorithm cannot easily find a good layout for the original graph, the same algorithm can find a good layout for a data set simplified using HITS for median ranking. A researcher looking at this data set can then visually see what protein clusters tend to interact with what other protein clusters, and what proteins seem to be unrelated.

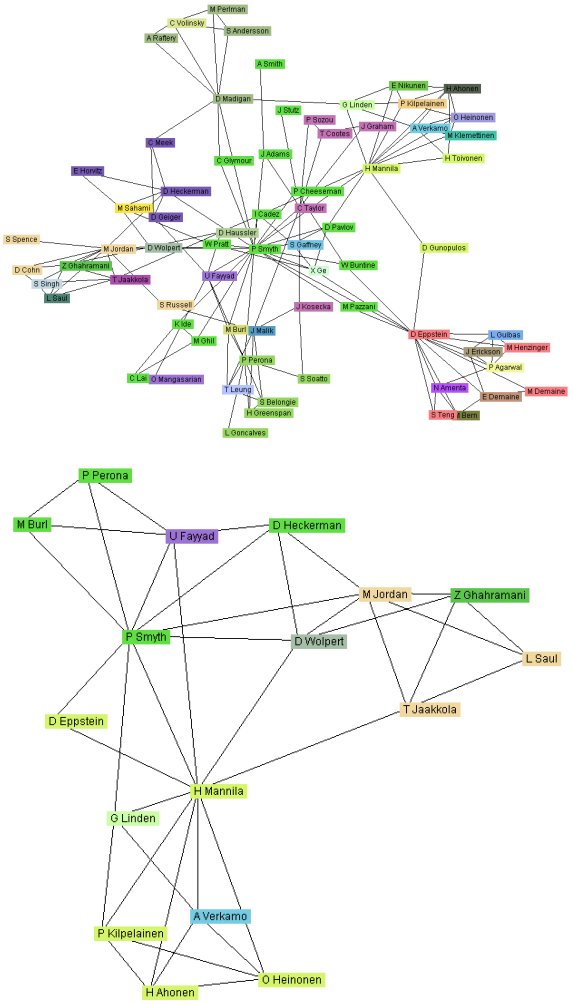


**Figure 3: Reduced Smyth.net – 71, 17 vertices**

# 5. SCALABILITY

Except for Betweenness-Centrality, which takes $O(V^2)$ time to rank $V$ vertices, each vertex ranking method takes $O(V)$ time to compute and $O(V)$ time to select the $p$ top ranked vertices. Growing $p$ clusters and constructing a multilevel mesh both take $O(V \log V)$ time. On a 933MHz Pentium III, constructing a multilevel mesh given a graph of six thousand vertices takes an average of five or six seconds.

In Table 5, we compare the scalability of Dijkstra's algorithm to our focused search (these experiments used a two-level structure, but more levels could be used to focus the search more tightly). We chose 512 random distance queries on the high energy physics publication data. We recorded the average number of vertices explored by Dijkstra's algorithm (column $b = 1$) as compared to focused search.

The number of vertices explored can be an order of magnitude smaller, depending on the average branching factor $b$. This is an important speedup if I/O dominates the cost of computation, as is common with massive data sets. The amount of time spent in the search was also reduced drastically whereas the error rates increased relatively slowly, as we increased the average branching factor.

| | b=1 | b=5 | b=10 | b=20 | b=40 |
|---|---|---|---|---|---|
| vertices explored | 2900 | 599 | 320 | 234 | 236 |
| approximation error | 0% | 5% | 12% | 18% | 22% |
| speedup factor | 1x | 5.5x | 13x | 18x | 16x |

**Table 5: hep-th branching factor comparison**

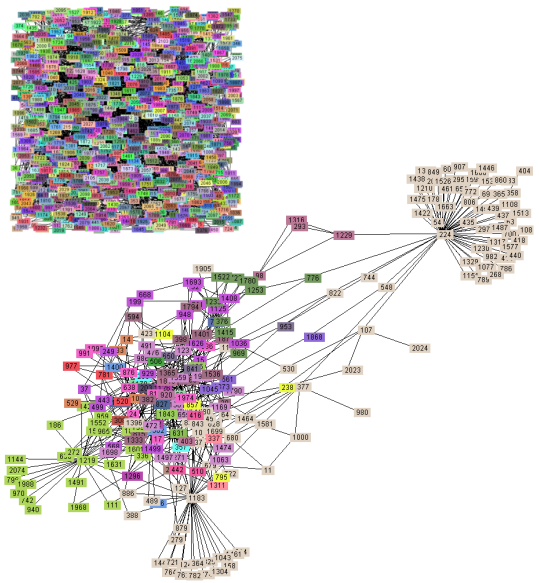

**Figure 4: protein-bo – 1458, 242 vertices**

## 6. CONCLUSION

The properties of scale-free networks have only been discovered in the past five years, and thus not much work has been done in modeling or analyzing such networks from an algorithmic or data mining perspective. However, since scale-free characteristics are found in many real-world graphs, we believe that it is an important problem to mine such networks. We developed algorithms that exploit these characteristics and applied new graph approximation techniques to large social networks and protein interaction graphs.

We introduced a form of graph clustering designed for large sparse graphs, specifically scale-free networks, based on geodesics and the idea of approximation clustering. We noted that the problem of graph data reduction is dual to the problem of graph clustering and described how to simplify graphs by contracting clusters to representative vertices.

Using this new framework, we described a new data structure called a multilevel mesh that approximates a graph system at multiple levels of detail. We showed that these multilevel structures are useful for visualizing large scale-free networks, understanding underlying graph patterns, and for speeding up computations on very large graphs.

## 7. FUTURE WORK

We developed algorithms for graph data reduction, but believe that our multilevel data structures could be applied toward other mining problems on scale-free networks. For example, we may want to exploit network structure when classifying graph data, performing visual data mining, or when looking for graph outliers that exhibit unusual connectivity patterns, which may correspond to terrorist activity in social networks [10] or fraudulent activity in a financial transaction graph. Other important problems include privacy-preserving data mining on social networks and the study of how such networks grow and change over time.

## 8. ACKNOWLEDGMENTS

We thank Sariel Har-Peled and Jeff Erickson for useful discussions. We also thank Mark Newman for the high energy theoretical physics publication data and Albert-László Barabási for the protein-protein interaction data set.

## 9. REFERENCES

[1] A. L. Barabási, R. Albert, H. Jeong, and G. Bianconi. Power-law distribution of the world wide web. *Science*, 287, 2000.

[2] B. Bollobás. *Random Graphs*. Cambridge University Press, second edition, 2001.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the Seventh International Conference on World Wide Web 7*, pages 107–117. Elsevier Science Publishers B. V., 1998.

[4] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 57–66. ACM Press, 2001.

[5] Z. Drezner and H. W. Hamacher. *Facility Location: Applications and Theory*. Springer, 2002.

[6] M. Garland. Multiresolution modeling: Survey & future opportunities. In *State of the Art Report*, pages 111–131. Eurographics, Sept. 1999.

[7] L. Holder, D. Cook, and S. Djoko. Substructure discovery in the subdue system. In *Proceedings of the Workshop on Knowledge Discovery in Databases*, pages 169–180, 1994.

[8] H. Jeong, S. P. Mason, A. L. Barabási, and Z. Oltvai. Lethality and centrality in protein networks. In *Nature*, volume 411, pages 41–42, 2001.

[9] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[10] V. Krebs. Mapping networks of terrorist cells. *Connections*, 24, 2001.

[11] B. Mirkin. *Mathematical Classification and Clustering*. Kluwer Academic Publishers, 1996.

[12] M. E. J. Newman. The structure and function of complex networks. In *SIAM Review*, volume 45, pages 167–256, 2003.

[13] M. E. J. Newman. Detecting community structure in networks. In *Eur. Phys. J. B.*, 2004.

[14] J. O'Madadhain, D. Fisher, S. White, and Y. Boey. The JUNG (Java Universal Network/Graph) framework. Technical report, UC Irvine, 2003.

[15] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 61–70. ACM Press, 2002.

[16] K. Schloegel, G. Karypis, and V. Kumar. Graph partitioning for high performance scientific simulations. In *CRPC Parallel Computing Handbook*. Morgan Kaufmann, 2000.

[17] T. Washio and H. Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, 2003.

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

[19] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 266–275. ACM Press, 2003.

[20] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Proc. 2002 Int. Conf. on Data Mining (ICDM'02)*, 2002.

[21] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *ACM SIGMOD Intl. Conf. on Management of Data*, pages 103–114, June 1996.