Proceedings of the 2007 IEEE Symposium on
Computational Intelligence and Data Mining (CIDM 2007)

More Efficient Classification of Web Content Using Graph Sampling

Chris Bennett
Department of Computer Science
University of Georgia
Athens, Georgia, USA 30602
bennett@cs.uga.edu

Abstract— **In mining information from very large graphs, processing time as well as system memory become computational bottlenecks as the properties of large graphs must be compared through each iteration of an algorithm. This is a particularly pronounced problem for complex properties. For example, distance metrics are used in many fundamental data mining algorithms including *k*-nearest neighbors for the classification task. Even the relatively efficient distance and similarity heuristics for large inputs, though, often require processing and memory well beyond linear with respect to the size of the input, and this rapidly becomes intractable with very large inputs. Complex properties such as the distance between two graphs can be extremely costly, but using samples of these large graphs to calculate the same properties proves to reduce memory requirements and processing time significantly without sacrificing quality of classification. Because the vast amount of web data is easily and robustly represented with graphs, a data reduction technique that preserves the accuracy of mining algorithms on such inputs is important. The sampling techniques presented here show that very large graphs of web content can be condensed into significantly smaller yet equally expressive graphs that lead to accurate but more efficient classification of web content.**

I. Introduction

Graphs provide a robust structure with which to model data for many computational problems. They are common inputs to problems in information systems, data mining, and image processing, and there is of course a rich theoretical background in graphs as well. Processing data in graph form can be expensive with respect to space and time efficiency, but very large graphs are not uncommon when modeling real world data. Graphs offer an especially rich representation for web-accessible content, and this work proposes and implements a novel use of random sampling to more efficiently classify large graphs created from such large real world corpora.

In this paper, graphs are created from the collected text of twenty newsgroup postings available in the UC Irvine Knowledge Discovery in Databases Archive [12], but the techniques apply easily to any large collections of text such as those created from web sites or emails. After some pre-processing, each of the newsgroup graphs initially has over 15,000 nodes and several times that many edges, and they require a great deal of computing resources when used in even basic data mining tasks. This research shows that random sampling can be used on these graphs in such a way that the samples themselves may be used as proxy inputs to common data mining tasks such as classification. Because structural properties and aggregate features of the input are maintained, the use of the condensed inputs improve the performance relative to both memory and speed without sacrifices in quality. Specifically, *k*-nearest neighbors using significantly smaller graphs results in the same classifications as the same algorithm used on the original inputs. Classification requires a means of determining a distance between graphs, and there are several ways to measure similarity between graphs. Various graph distance metrics which are used in common clustering and classification algorithms are listed, but due to space constraints only the results from experiments using the maximum common subgraph metric are used to show that the results on the sample graphs are of the same quality as the much larger original input graphs.

It is important to distinguish between the effectiveness of the particular classification methods used and the effectiveness of the sampling method. This work is concerned with reproducing the same results with the samples as with the much larger original graphs, and this means that misclassifications should be seen as well. The focus is not necessarily on improving the accuracy of the classification method but rather on the accuracy of the sample as compared to the original input. Any improvements in accuracy for a given method should transfer to the sample inputs but with the advantage of reduced processing.

II. Related Work

Sampling is widely used in computation, but it is typically associated with finding a representative subset of inputs from a large number of inputs. Sampling can also be used to reduce a single large input to a more manageable yet still representative structure for use in computations. Previous work [2] shows the effectiveness of random sampling in reducing the space complexity of computational problems while maintaining the relevant structural features of large inputs. Such properties are useful in several areas. For example, the algorithm selection problem focuses on selecting the best algorithm among a collection of solutions whose efficiencies depend on features of the input [9]. In sorting, different sorting algorithms perform better or worse depending on the prior relative ordering of the

input [8], [5]. For graphs, the efficiency of different algorithms for finding the minimum spanning tree (for example, Kruskal's and Prim's algorithms) can vary according to the density of the graph.

Other related work examines the effectiveness in sampling in computation as well. Leskovec and Faloutsos test the ability of multiple sampling schemes to estimate individual features such as in- and out-degree of nodes from the original input [7]. They show that indeed sampling can be used to preserve context-independent features of many large graphs that model real world data such as citation networks. Other recent research using novel sampling tecniques has been used to re-create the estimated coverage of a search engine's index [1]. The preservation of other graph properties from network graphs with relatively small samples is studied in [6].

There has also been work on using graphs in web content mining because of their ability to represent relationships between objects or elements in a set. This additional representational ability, though, often comes with a computational cost. Schenker et al. explore the use of graphs to overcome the inability of vector models to represent structural information [10]. They show that graphs can be effective structures for web content mining tasks such as classification and clustering, but the computational overhead can be prohibitive for large graphs. Other techniques for dimensionality reduction include latent semantic indexing [4] that rely on the use of singular value decomposition to condense the input, and others have shown that such techniques can indeed provide as good or better performance for data mining tasks [11]. However, this work differs in that it attempts to maintain the rich structure inherent in graphs for use in classification.

Sampling of course has been used to choose from a large number of inputs to narrow the complexity of the input space in many data mining applications, but little work has focused on using sampling to reduce the complexity of a single or many very large inputs to such a task. This research shows that sampling can also be used with large graphs to classify data more efficiently but just as accurately as with the original inputs. Using data from the text from a collection of twenty newsgroups postings [12], large graphs are created and classified into different groups using a traditional k-nearest neighbors algorithm using a distance metric known as the maximum common subgraph. Samples of these large graphs are then processed in the same way, and results are provided that indicate samples indeed provide very accurate results with significantly smaller processing and memory requirements.

III. Sampling and Graphs

Large graphs as models of real world problems are a natural result of their ability to represent rich relationships between objects as well as the constant increase in available machine-processable information. Processing such large graphs becomes very costly, particularly for problems whose solutions scale polynomially (or worse) with the size of the input. Heuristics for reducing the search space are a common approach, and reducing the input to a more compact representation also can be critical in designing a tractable solution. Data reduction must not come at the expense of accuracy of the model, though. This section addresses accurately estimating such distance or similarity metrics for large graphs through sampling.

A. Large Graphs and Their Features

Graphs are commonly defined as a structure of two sets – nodes and the edges that connect them. There are of course as many variations on this basic idea as there are problems to which to apply them, and these can be found in any introductory graph theory text. Graphs are commonly used not only to store objects (nodes) but the relationships between them as well (edges). Many data structures have structural features that make a particular instance unique in a sense, and graphs are no different. Given a particular graph, such features as the density and edge probability are context-independent ways to describe the data in the aggregate. In addition, there are problem- and context-specific features of graphs that reveal much about a particular instance and its relationship with the problem. These features can uniquely identify the input, and they can affect the performance of different algorithms used to solve the problem. These features are also frequently the output of a problem themselves. For example, the similarity (or distance) between two inputs is required for many data mining processes such as clustering and classification. It is these features on which this work focuses.

Experiments on randomly generated graphs are useful to show that abstract features that are not context specific are maintained through sampling. Experiments on real data, however, show that features unique to a particular application or context can also be accurately estimated with an appropriate sampling technique. Large volumes of data naturally result in large computational representations, and these structures are rich in features as well as noise. Data to create such large graphs are readily available in various domains such as network traffic analysis, image processing, and the mining of data from large text corpora [10], and this work uses sampling to filter noise out of large graphs created from web content for more efficient classification.

B. Sampling Graphs

There are a variety of ways to sample graphs. While some focus on the set of nodes, others focus on the edge set. Random walks and its variations are also often used to explore graphs to create a sample. Each method has its own advantages and disadvantages, and the best method often depends on the application or data itself. In random node selection, random nodes are selected until a target sample size is reached. Variations often differ on exactly which edges to include from the a node's adjacency list. In random edge selection, random edges are added to a target sample's edge list until a required size is reached. Both of these techniques have their biases toward or against certain kinds of nodes, though. Random walk-based sampling techniques often select a random node

from which to start a walk across a graph by selecting edges to traverse from the current node's adjacency list. Variations often manipulate various probabilities that influence the chance of returning to the start node or a new random node, for example. In this paper, experiments use a hybrid edge-node selection sampling method to create samples. Its details are discussed in Section 5. The ability of this sampling technique to estimate aggregate features such as the relative distance between graphs will be evaluated for the classification task.

IV. Graph Similarity

Measuring how alike two graphs are is a computationally complex problem. If two graphs are considered isomorphic, we might say that they are the same graph with different labels. If they are not exactly isomorphic, though, we might want to describe how close they are to being exactly alike (or unalike). Such notions of relative proximity are very useful in tasks such as classification and clustering. Though of course we ideally want a highly accurate but computationally inexpensive distance metric, there exists as always a tradeoff. Several such measures are proposed in [10]. Our experiments use one particular metric for similarity calculations, and it then addresses the question of whether the same technique produces the same relative results when used on samples of the original inputs.

A. Graph Distance Metrics

Measuring the similarity between graphs provides a notion of how far apart or how close two graphs are, but they can be most useful when several such graphs are being compared. Schenker et al. list several such graph similarity techniques [10] including but not limited to:

- Graph and subgraph isomorphism
- Graph edit distance
- Maximum common subgraph
- Minimum common supergraph
- Probabilistic
- Distance preservation

The authors provide a thorough treatment of the advantages and disadvantages of each technique. In addition, they also list several ways to calculate the means and medians of graphs for clustering tasks. Experiments discussed in this work use the maximum common subgraph distance measure.

B. Maximum Common Subgraph

The maximum common subgraph technique attempts to find the largest subgraph that two graphs have in common, and it is related to the graph edit distance [3]. Given graphs G_1 and G_2, we define *mcs(G_1, G_2)* as the maximum common subgraph of G_1 and G_2 if it is a subgraph of G_1, it is a subgraph of G_2, and there exists no other subgraph that is contained in both G_1 and G_2 that is larger. With unlabeled graphs, such a measure would be very inefficient. With labeled graphs, however, the maximum common subgraph is computationally manageable because the labels significantly reduce the possible matchings. Space and time requirements are still very high with very large graphs, however. The algorithm used in the experiments is as follows:

Algorithm 1: Maximum Common Subgraph

```
Input: Graphs G1 and G2
Output: Maximum Common Subgraph of G1 and G2
for edge e ∈ G1 do
    if e ∈ G2 then
        add e to g
    end
end
return g
```

Table 1 shows distances between a subset of the graphs to all other graphs created from the original text of the twenty newsgroups. Not all graphs from the collection of twenty are listed due to space constraints, but those listed in the table are representative of those not listed. The closest graph (which is not itself) by maximum common subgraph is listed in bold for each column. The distances can have a range from 0 to 1. Here, a distance of 0 means the graphs are the same while a distance of 1 means that the graphs have no edges (with the same from and to nodes) in common. These distances between newsgroups make intuitive sense. That is, we would expect newsgroups about computers to be closer to each other than to those about religion, for example. Details about how these graphs are created are discussed in the next section. With large graphs, these calculations can rapidly exceed the computational limits of a particular application. Sampling provides a way to reduce the size of the inputs significantly while still preserving its aggregate structure so that relative ordering with respect to similarity between all pairs of graphs is maintained given a set of inputs. This results in large gains in performance for data mining tasks with little or no sacrifice in quality.

V. Sampling for Similarity

The effectiveness of graph sampling methods is often dependent on the data being represented. We examine in more detail a hybrid random edge-node selection method for creating a subgraph of an input, but there exist several sampling methods which may be used to varying effectiveness for a given application. We create samples using the hybrid edge-node selection procedure, and we compare the relative ordering of distance metrics as calculated by the maximum common subgraph in the original inputs as well as the sample graphs.

A. Hybrid Edge-Node Random Sampling

A randomly selected subgraph *g'* of size *s* of some graph *G* can be created by randomly selecting *s* nodes and adding them to *g'*. Variations might select all or part of each node's adjacency list to add to *g'* as well. Alternatively, edges can be randomly selected from *G's* edge list to add to *g'* along with

	Atheism	Autos	CompIBMHardware	CompWindowsMisc	Motorcycles	PoliticsMideast	PoliticsMisc	Religion	SportsHockey
Atheism	-	0.79	0.83	0.88	0.79	0.76	0.74	**0.59**	0.80
Autos	0.79	-	0.78	0.87	**0.74**	0.82	0.80	0.80	0.80
CompGraphics	0.80	0.81	0.79	**0.84**	0.82	0.83	0.81	0.81	0.82
CompIBMHardware	0.83	0.78	-	0.84	0.79	0.86	0.84	0.84	0.83
CompMacHardware	0.83	0.78	**0.73**	0.86	0.78	0.86	0.84	0.84	0.83
CompWindowsMisc	0.88	0.87	0.84	-	0.88	0.87	0.86	0.87	0.88
CompWindowsX	0.84	0.85	0.83	0.85	0.85	0.85	0.83	0.84	0.86
ForSale	0.84	0.78	0.77	0.88	0.80	0.86	0.85	0.85	0.83
Motorcycles	0.79	**0.74**	0.79	0.88	-	0.83	0.81	0.81	0.81
PoliticsGuns	0.73	0.78	0.82	0.87	0.78	0.74	0.67	0.69	0.78
PoliticsMideast	0.76	0.82	0.86	0.87	0.83	-	0.67	0.75	0.82
PoliticsMisc	0.74	0.80	0.84	0.86	0.81	**0.67**	-	0.64	0.80
ReligionChristian	0.70	0.80	0.83	0.87	0.80	0.75	0.74	0.69	0.80
Religion	**0.59**	0.80	0.84	0.87	0.81	0.75	**0.64**	-	0.81
ScienceCrypt	0.75	0.79	0.81	0.85	0.80	0.78	0.75	0.75	0.81
ScienceElectronics	0.81	0.75	0.75	0.86	0.77	0.84	0.82	0.82	0.82
ScienceMed	0.77	0.79	0.82	0.86	0.80	0.78	0.75	0.75	0.81
ScienceSpace	0.77	0.79	0.82	0.86	0.80	0.77	0.75	0.75	0.80
SportsBaseball	0.80	0.77	0.81	0.88	0.77	0.83	0.80	0.81	**0.77**
SportsHockey	0.80	0.80	0.83	0.88	0.81	0.82	0.80	0.81	-

TABLE I
DISTANCES BETWEEN ORIGINAL INPUTS USING MAX COMMON SUBGRAPH.

its "to" and "from" nodes. Random node or edge selection to create samples has certain biases toward or against nodes with higher or lower in and out degrees, and some of these can be avoided through a hybrid approach. The following algorithm creates a subgraph *g'* of *G* of size *s* by randomly selecting edges, adding nodes, and then processing the adjacency lists of each node previously added.

This algorithm performs well for many types of graphs for maintaining general structure. Particular applications may require other techniques such as a random walk sampling. For example, an application may require that the sample be a connected component of the original input. In these experiments, however, no such requirements exist. In this case, other techniques may work better, and tests with other sampling schemes have had similar results.

B. Data

Randomly generated artificial graphs provide complete control over features of the graph at the individual node and edge level as well as the aggregate level, but graphs representing real world data provide motivation for application of sampling techniques [2], [7]. A collection of newsgroup postings from the UC-Irvine Knowledge Discovery in Databases archive are used [12]. These files contain 20,000 messages from twenty newsgroups (1,000 from each newsgroup). From these,

***Algorithm 2**: Hybrid Random Node-Edge Graph Sampling*

Input: Graph G, Subgraph graph size s
Output: Subgraph g' of G
while $\mid g' \mid \leq s$ **do**
  select random edge e from G
  if $e \notin g'$ **then**
    add source node to g'
    add destination node to g'
  end
end
for *all nodes* $n \in g'$**do**
  for *edge* $e \in n$*'s adjacency list* **do**
    if *destination node* $\notin g'$ **then**
      remove e
    end
  end
end
return g'

twenty graphs are constructed based on the words contained in the messages and their proximity to each other. Stop word removal is performed to reduce the size of the graph and remove extremely common words from all groups. Other preprocessing techniques such as stemming are possible, but some noise is left in the graph representation by performing only basic stop word removal to test how robust the sampling techniques can be. Directed edges are created between words within three positions of each other, but this parameter can be adjusted to create more or less dense graphs. Sample sizes of 25, 40, and 50% are used in experiments, but only the results from 50% samples are discussed here.

C. Experiments

For the experiments, the original graphs are processed as pairs to compute distance according to the maximum common subgraph of each pair. The distance is calculated as the ratio of the size of the maximum common subgraph to the size of the larger of G_1 and G_2. If, for example, the size of graph G_1 is 100 and graph G_2 is 150, and the size of the maximum common subgraph is 75, the distance between G_1 and G_2 is 0.5. Several variations on this calculation exist as detailed in [10]. Then a sample of each graph is taken, and the distance between each pair of samples is calculated. Comparisons are made between the predicted order of closest to furthest from the samples to the actual distances calculated from the original inputs.

D. Results

Table 2 lists the changes in relative rankings (for the same newsgroups used in Table 1) from a 50% sample to the original input. For example, if the original input had the atheism and religion newsgroups as closest to each other and the sample had them closest together as well, the change in proximity ranking from input to sample is 0. If the sample had another group closer to religion than atheism (say atheism was the second closest), the change in the relative ranking is -1.

	Atheism	Autos	CmplBMHardware	CompWindowsMisc	Motorcycles	PoliticsMideast	PoliticsMisc	Religion	SportsHockey
Atheism	0	-2	-2	-1	-1	0	-1	0	0
Autos	0	0	-1	0	0	1	0	0	-1
CompGraphics	3	1	1	0	1	3	3	3	2
CmplBMHardware	0	1	0	0	0	1	0	1	-1
CompMacHardware	0	-1	0	0	0	-1	0	-2	-1
CompWindowsMisc	0	0	0	0	0	0	0	0	0
CompWindowsX	1	0	1	0	0	0	0	1	0
ForSale	-1	0	0	1	1	0	0	0	3
Motorcycles	-2	0	0	0	0	-1	-1	-2	-1
PoliticsGuns	0	1	0	0	0	0	0	0	0
PoliticsMideast	0	0	0	0	0	0	0	0	-3
PoliticsMisc	0	1	0	1	0	0	0	0	1
ReligionChristian	0	0	2	0	0	0	0	0	-2
Religion	0	0	1	0	0	0	0	0	0
ScienceCrypt	0	-1	1	0	-1	0	0	0	0
ScienceElectronics	1	0	0	0	0	0	0	1	1
ScienceMed	0	1	0	0	2	0	1	1	0
ScienceSpace	0	1	0	-1	-1	0	0	-1	2
SportsBaseball	0	-1	-1	1	0	0	0	0	0
SportsHockey	-2	-1	-2	-1	-1	-3	-2	-2	0

TABLE II
CHANGE IN RELATIVE RANKING OF DISTANCE FROM 50% SAMPLE TO ORIGINAL INPUT.

We see from the results in the table that samples taken from the original graphs do an excellent job of maintaining relative ordering with respect to the distance metric used. The variations that do exist are minor, and they are attributed to the original distances between certain groups being clustered into close groups initially. We might expect then that a sample might flip an ordering here and there, but the same neighborhoods exist in the rankings. There are no large variations between the original rankings and the rankings of the sample, showing that the samples are similarly structured despite being significantly smaller. Data from exhaustive experiments using sample sizes of 25, 40, and 50% are available but not included due to space constraints. As expected, smaller samples lead to more variation, but results are still positive for smaller sizes. Having shown that samples preserve graph structure, we can now use these similarity measures in applications. As we will see in the next section, the distances between the samples result in the same classifications as the distances between the original inputs when used in a *k*-nearest neighbors algorithm.

VI. CLASSIFICATION WITH SAMPLES

Classification tasks typically attempt to classify a previously unseen input into a previously defined class. *K*-nearest neighbors (knn) is a common algorithm that classifies an input based on the classification of its *k*-closest neighbors as determined by some distance metric. In these experiments, maximum common subgraph provides the measure for similarity between two graphs, and various neighborhood sizes are used. Sample sizes of 25, 40 and 50% are used with k set to 3, 4, and 5. Results for the 50% with k=3 are discussed specifically.

A. Experiments

For these experiments, five classification groups are used in which to categorize each newsgroup. These author-created labels are religion, politics, computer, science, and other. These are of course arbitrary, and other current work includes using similarly created samples for clustering tasks to determine whether samples cluster into the same arbitrary neighborhoods as the original inputs. Other experiments with other classification types have been done, and the results are equally accurate. The author has assigned a default classification to each newsgroup, and the initial twenty graphs are classified according to a vote from the three closest neighbors (k=3) as measured by the maximum common subgraph. Then samples are created for each graph, and the same process is repeated for all pairs of sample graphs. The results from experiments with neighborhood size of three with a sample size of 50% are discussed.

B. Results

The results for classification for the 50% sample of each of the original input graphs are shown in Table 3. For each newsgroup, its author-assigned classification is listed as well as its *k*-nearest neighbor classification from the original inputs. The other two columns indicate whether the algorithm accurately predicted the same group as the author-assigned category using the original inputs as well as the samples from these inputs. The results indicate that the algorithm misclassifies some of the graphs using both the original graphs and the samples. Several improvements could be implemented to improve the accuracy incrementally, but the focus is not on whether the classifications of the samples are correct but rather on whether the classifications are the same as the original large graphs. Indeed, the samples predict the same classifications as the *k*-nearest neighbor run on the original inputs. Any errors or successes made by the algorithm using the original inputs are repeated for every sample.

These results and data from other experiments using different sample sizes and classifications lead to the conclusion that sampling these large inputs can and do result in similarly structured graphs, and the maximum common subgraph between these samples produces the same general neighborhoods of proximity in the collection of graphs. By using the samples in our classification rather the original, we can significantly reduce the run time for classification as well as the space requirements by more than half. Additional experiments show that as one would expect, smaller sample sizes have more variation from the classification of the originals. Even systematically reducing the input by 75%, though, produces efficient and relatively accurate results.

50%	Newsgroup	Category	KNN	Right?	Same?
	Atheism	Religion	Religion	1	1
	Autos	Other	Other	1	1
	CompGraphics	Computer	Science	0	1
	CompIBMHardware	Computer	Computer	1	1
	CompMacHardware	Computer	Computer	1	1
	CompWindowsMisc	Computer	Computer	1	1
	CompWindowsX	Computer	Science	0	1
	ForSale	Other	Computer	0	1
	Motorcycles	Other	Other	1	1
	PoliticsGuns	Politics	Politics	1	1
	PoliticsMideast	Politics	Politics	1	1
	PoliticsMisc	Politics	Politics	1	1
	ReligionChristian	Religion	Religion	1	1
	Religion	Religion	Religion	1	1
	ScienceCrypt	Science	Politics	0	1
	ScienceElectronics	Science	Computer	0	1
	ScienceMed	Science	Politics	0	1
	ScienceSpace	Science	Politics	0	1
	SportsBaseball	Other	Other	1	1
	SportsHockey	Other	Other	1	1

TABLE III
50% SAMPLE CLASSIFICATION COMPARED TO FULL INPUT CLASSIFICATION.

VII. FUTURE WORK

Reducing large inputs such as graphs to representative samples has many applications. Current work is focusing on using these sampling techniques on other data mining tasks such as clustering. Several notions of the means and medians of graphs exist, and it would be interesting to explore the effects of sampling on them. By exploiting the structure preservation that sampling has been shown to provide, there exist abundant opportunities not only for improved efficiency in existing tasks but for data reduction tasks for very large inputs as well. Future efforts will explore uses in other web-accessible content such as large scale website content and social networks. In addition, other types of data such as time-series flows from sensor networks might benefit from data reduction techniques.

VIII. CONCLUSIONS

This work has shown that the use of random samples of large inputs to data mining processes can successfully reduce the time and memory requirements while providing the same results for tasks such as classification. The k-nearest neighbors algorithm produces the same classifications whether the inputs were the original graph created from twenty collections of newsgroup postings or random samples taken from these graphs. These improvements in efficiency are due to the ability of sampling to retain the properties of large inputs such as graphs created from web content. As inputs to mining tasks grow larger and larger, such data reduction techniques become more and more important to maintain efficient computation.

REFERENCES

[1] Z. Bar-Yossef and M. Gurevich. Random Sampling from a Search Engine's Index. In *Proceedings of the 15th International Conference on World Wide Web*. ACM Press, New York, NY, 367-376. 2006.

[2] C. Bennett and W. D. Potter. Input Sampling and Sorting Algorithm Selection. To appear in a special issue of the *CSDA Journal on Model Selection*. 2005.

[3] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Patter Recognition Letters*, 18:689-694, 1997.

[4] S. Dumais, T. Furnas, T. Landaur, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391-407, 1990.

[5] Estivill-Castro, Vladmir and Derick Wood. A Survey of Adaptive Sorting Algorithms. *ACM Computing Survey* v. 24, ACM Press, New York, NY, USA, 441-476, 2002.

[6] V. Krishnamurthy, M. Faloutsos, M. Chrobak, L. Lao, J.H. Cui, and A.G. Percus. Reducing Large Internet Topologies for Faster Simulations. In *Networking*, 2005.

[7] J. Leskovec and C. Faloutsos. Sampling from Large Graphs, *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 631-636, 2006.

[8] H. Mannila. Measures of presortedness and optimal sorting algorithms. *IEEE Transactions on Computers 34* , 318-325, 1985.

[9] J. R. Rice. The Algorithm Selection Problem. *Advances in Computers*, 15:65-118, 1976.

[10] A. Schenker, H. Bunke, M. Last, and A. Kandel. *Graph-Theoretic Techniques for Web Content Mining* (Machine Perception and Artificial Intelligence). Series in Machine Perception and Artificial Intelligence). World Scientific Publishing Co., Inc. River Edge, NJ, USA. 2005.

[11] H. Schutze and C. Silverstein. Projections for efficient document clustering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 74-81, 1997.

[12] UC-Irvine KDD Repository, http://kdd.ics.uci.edu/. Last checked October 20th, 2006.