Graph Clustering with Network Structure Indices

Matthew J. Rattigan RATTIGAN@CS.UMASS.EDU
Marc Maier MAIER@CS.UMASS.EDU
David Jensen JENSEN@CS.UMASS.EDU
Knowledge Discovery Laboratory, Department of Computer Science, University of Massachusetts Amherst

Abstract

Graph clustering has become ubiquitous in the study of relational data sets. We examine two simple algorithms: a new graphical adaptation of the k-medoids algorithm and the Girvan-Newman method based on edge betweenness centrality. We show that they can be effective at discovering the latent groups or communities that are defined by the link structure of a graph. However, both approaches rely on prohibitively expensive computations, given the size of modern relational data sets. Network structure indices (NSIs) are a proven technique for indexing network structure and efficiently finding short paths. We show how incorporating NSIs into these graph clustering algorithms can overcome these complexity limitations. We also present promising quantitative and qualitative evaluations of the modified algorithms on synthetic and real data sets.

1. Introduction

Clustering data is a fundamental task in machine learning. Given a set of data instances, the goal is to group them in a meaningful way, with the interpretation of the grouping dictated by the domain. In the context of *relational* data sets — that is, data whose instances are connected by a link structure representing domain-specific relationships or statistical dependency — the clustering task becomes a means for identifying communities within networks.

For example, in the bibliographic domain, we find networks of scientific papers. Interpreted as a graph, vertices (papers) are connected by an edge when one cites the other. Given a specific paper (or group of papers), one may try to find out more about the subject matter by pouring through the works cited, and perhaps the works they cite as well. However, for a sufficiently large network, the number of papers to investigate quickly becomes overwhelming. By clustering the graph, we can identify the community of relevant works surrounding the paper in question. In the sections that follow, we discuss methods for clustering such graphs into groups that are solely determined by the network structure (e.g., co-star relations between actors or citations among scientific papers).

Some of the simplest approaches to graph clustering are also very effective. We consider two algorithms: a graphical version of the k-medoids data clustering algorithm (Kaufman & Rousseeuw, 1990) and the Girvan-Newman algorithm (2002). While both techniques perform well, they are computationally expensive to the point of intractibility when run on even moderate-size relational data sets. Using the indexing methods described by Rattigan, Maier, and Jensen (2006), we can drastically reduce the computational complexity of these algorithms. Surprisingly, this increase in scalability does not hinder performance.

2. Graph clustering algorithms

2.1. Evaluating clustering performance

Before examining the details of the graph clustering algorithms, we introduce a framework for analyzing and evaluating clustering performance. We evaluate candidate algorithms on randomly generated uniform clustered graphs (Brandes et al., 2003; Delling et al., 2006), whose link structure defines communities of nodes. The data generator constructs a graph $G = (V, E)$ as follows: Given a set of nodes V, randomly assign each node to one of j clusters, such that the final cluster sizes are normally distributed about mean μ. Pairs of nodes in the same cluster are connected by a link with probability p_{in}, and pairs of

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

nodes in different clusters are linked with probability p_{out}. By varying these parameters, we can adjust the expected number of intra- and inter-cluster links for each node, and in turn, control the level of separation between clusters. The degree of difficulty for the clustering task can be captured by the expected inter-cluster linkage for each node $v \in V$:

$$\text{inter-cluster linkage} = E\left[\frac{\text{degree}_{\text{inter}}(v)}{\text{degree}(v)}\right]$$

where degree(v) is the total number of links incident to v, and degree$_{inter}(v)$ is the number of links connecting v to nodes outside its cluster.

We gauge clustering performance with two measures. Pairwise intra-cluster accuracy (A_{intra}) is the proportion of all pairs of nodes in the same cluster that are predicted to be in the same clusters. Similarly, pairwise inter-cluster accuracy (A_{inter}) is the proportion of all pairs of nodes in different clusters that are predicted to be in separate cluster. These measures are formally defined as follows:

$$A_{\text{intra}} = \frac{\sum_{u,v \in V_{pair-intra}} f(u,v)}{|V_{intra}|}$$

$$A_{\text{inter}} = \frac{\sum_{u,v \in V_{pair-inter}} (1 - f(u,v))}{|V_{inter}|}$$

where $f(u,v)$ is 1 when u an v are in the same predicted cluser and 0 otherwise, and $V_{pair-intra}$ and $V_{pair-inter}$ are the sets of intra- and inter-cluster pairs of nodes, respectively. Both values measure our effectiveness at reproducing the natural clusters found in the graph. Another measure of clustering accuracy, the Rand Index (Rand, 1971), can also be expressed in these terms:

$$\text{Rand}R = \frac{A_{intra} * V_{pair-intra} + A_{inter} * V_{pair-intra}}{n}$$

For the results reported in this paper, the accuracy measures are approximated by sampling 10,000 pairs of nodes.

2.2. Graph *k*-medoids

The k-medoids algorithm (Kaufman & Rousseeuw, 1990) is fairly simple, and can be though of as a discrete version of the ubiquitous k-means data clustering technique (MacQueen, 1967). The inputs to the algorithm are k, the number of clusters to form, and $D : (x, y) \rightarrow \Re$, a distance measure that maps pairs of instances to a real value. The procedure is as follows: (1) randomly designate k instances to serve as "seeds" for the k clusters; (2) assign the remaining data points to the cluster of the nearest seed using D; (3) calculate the medoid of each cluster; and 4) repeat steps 2 and 3 using the medoids as seeds until the clusters stabilize.

We extend the above procedure to the network domain as the *graph* k-*medoids* algorithm. In graphs, we have an intuitive measure for D: the geodesic distance, or number of "hops" between nodes. As in conventional k-medoids, we initialize our clusters by randomly selecting k data points (in this case nodes in the graph) as seeds and assigning all nodes to the cluster of the nearest seed node. We choose medoids by computing the local closeness centrality (Freeman, 1979) among the nodes in each cluster and selecting the node with the greatest closeness score. This process terminates when the cluster medoids stabilize.

There are subtle differences between the graphical version of k-medoids and its data-clustering counterpart. For example, graph distance is highly sensitive to the edges that exist in the graph. Adding a single "shortcut" link to a graph can reduce the graph diameter, altering the graph distance between many pairs of nodes. Additionally, since graph distances are integers, it is common for nodes to be equidistant to several cluster medoids. We resolve conflicts by randomly selecting a cluster; however, this can result in clusterings that do not converge. We consider a clustering to be stable if the number of cluster medoids that change between iterations is below a certain threshold, typically 1-3%. It is possible for several nodes to have identical closeness centrality, forcing the algorithm to select medoids randomly.

To evaluate the clustering ability of graph k-medoids, we generated 10 data sets of 1,000 nodes with mean cluster sizes of 10, 20, and 50 (and standard deviations of 2, 4, and 10, respectively), and averaged the performance of 10 runs of graph k-medoids per structure to reduce variance. In Figure 1, we see that A_{intra} decreases as inter-cluster linkage increases, becoming unacceptably low for even modestly challenging clustering tasks. Additionally, as the expected size of clusters increases, the performance continues to drop. In contrast, graph k-medoids scored adequately in terms of inter-cluster accuracy, always maintaining A_{inter} scores above 0.99, 0.98, and 0.95 for clusters of size 10, 20, and 50, respectively.

Upon examination, most of the errors associated with graph k-medoids involve nodes lying on the periphery of clusters, resulting from the use of the integral-valued graph distance within an algorithm that prefers continuity. Figure 2 illustrates this primary source of error. In this example, node A's connections clearly place it in the light shaded cluster. However, A is directly linked to two medoids, and graph k-medoids

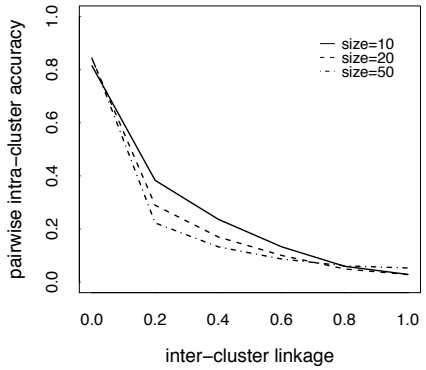

Figure 1. Clustering performance of graph k-medoids for synthetically generated data sets with mean cluster sizes of 10, 20, and 50. As inter-cluster linkage increases, the structural separation between clusters decreases, and the clustering task becomes more difficult. The maximum variance of any point is 0.0017; results following exhibit similar minimal variance as well. We omit pairwise inter-cluster accuracy, which for all runs ranged between 0.95 and 0.99.

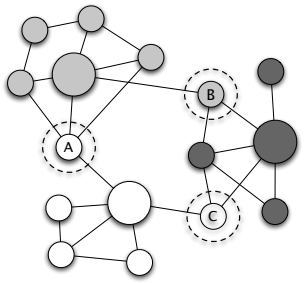

Figure 2. An illustrative example of typical graph k-medoids clustering errors. Nodes A, B, and C are equidistant from multiple medoids (enlarged nodes) and are erroneously clustered.

randomly (and incorrectly) assigns A to the white cluster. Nodes B and C are incorrectly clustered in a similar manner. For this small example, the clustering has $A_{intra} = 0.611$ and $A_{inter} = 0.845$.

Given the nature of the clustering errors depicted in Figure 2, we can improve performance through a novel post-processing step called *modal reassignment* (MRA). We randomly iterate through the nodes of the graph, and examine the cluster membership of the immediate neighbors of each one. Then, we assign that node to the most common cluster among the node's set of neighbors. While this additional step takes $O(|E|)$ operations, the performance improvements are dramatic. Figure 3 compares graph k-medoids with and without MRA on synthetic data. This revision to the clustering algorithm is entirely separate from the graph k-medoids process and can be applied to a graph clustering generated by any algorithm. Furthermore, this simple technique illustrates the power of utilizing network structure in ways that are not applicable in traditional independent and identically distributed data contexts.

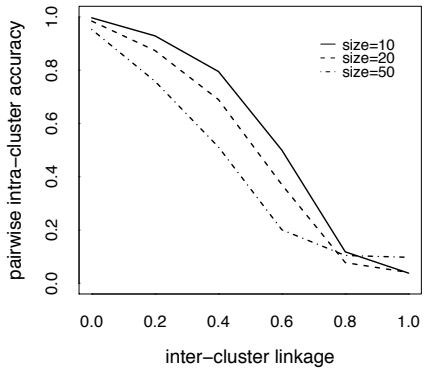

Figure 3. Clustering performance of k-medoids using modal reassignment

2.3. The Girvan-Newman algorithm

The Girvan-Newman algorithm (Girvan & Newman, 2002) is a divisive clustering technique based on the concept of *edge betweenness centrality*. Betweenness centrality is the measure of the proportion of shortest paths between nodes that pass through a particular link. Formally, betweenness is defined for each edge $e \in E$ as:

$$B(e) = \sum_{u,v \in V} \frac{g_e(u,v)}{g(u,v)},$$

where $g(u,v)$ is the total number of geodesic paths between nodes u and v, and $g_e(u,v)$ is the number of geodesic paths between u and v that pass through e. The algorithm ranks the edges in the graph by their betweenness and removes the edge with the highest score. Betweenness is then re-calculated on the modified graph, and the process is repeated. At each step, the set of connected components of the graph is considered a clustering. If the desired number of clusters is known *a priori* (as with k-medoids), we halt when the desired number of components (clusters) is obtained. The Girvan-Newman algorithm has been shown to perform well on a variety of graph clustering tasks (Newman, 2004a), but as we describe below, its complexity can severely limit its applicability.

3. Network structure indices

The main problem with graph k-medoids as described above is its complexity. As relational data sets become larger, the scalability of the algorithm becomes

an issue. Calculating (and storing) pairwise node distances (an $O(|V|^3)$ operation requiring $O(|V|^2)$ space) may be intractable for large graphs[1]. Similarly, the Girvan-Newman algorithm can be hampered by complexity. Calculating the edge betweenness for the links in a graph is a $O(|V||E|)$ operation, and the problem can become intractable for graphs with many nodes. Tyler, Wilkinson, and Huberman introduced a sampling-based approach to estimating betweenness centrality (2005); however, even with this improvement, finding the geodesic paths through the graph to calculate betweenness can be a prohibitively expensive set of operations. However, these limitations can be alleviated through the use of a *network structure index*.

A network structure index (NSI) is a scalable technique for capturing graph structure (Rattigan et al., 2006). The index consists of a set of node annotations combined with a distance measure. NSIs enable fast approximation of graph distances and can be paired with a search algorithm to efficiently discover short paths between nodes in the graph. For use in the clustering algorithms described in Section 2, we employed a distance to zone (DTZ) index (Rattigan et al., 2006). The DTZ indexing process creates d independent sets of random partitions (referred to as dimensions) by stochastically flooding the graph. Each dimension consists of z random partitions (referred to as "zones"). DTZ's annotations store the distances between each node and all zones across each dimension. Figure 4 illustrates an NSI with a single dimension and three zones. The distance between two nodes u and v is defined as:

$$D_{DTZ}(u,v) = \sum_d \text{dist}_d(u, \text{zone}(v)) + \text{dist}_d(v, \text{zone}(u))$$

where $\text{dist}_d(u, \text{zone}(v))$ is the length of the shortest path between u and the closest node in the same zone as v. Creating the DTZ index requires $O(|E|zd)$ time and $O(|V|zd)$ space. Since typical values of z and d are $\ll |V|$, a DTZ index can be created and stored in a fraction of the time and space it takes to calculate exact graph distances for all pairs of nodes in the graph.

Both graph k-medoids and the Girvan-Newman methods can be modified to utilize an NSI. For graph k-medoids, we substitute the DTZ annotation distance for exact graph distance when assigning nodes to clusters and calculating closeness centrality to determine medoids. As illustrated in Figure 5, the performance gains are dramatic.

[1]To be precise, the most efficient algorithm is currently $O(|V|^{2.376})$ (Coppersmith & Winograd, 1987)

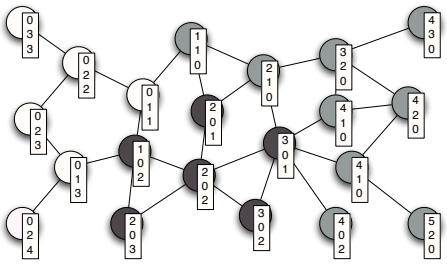

Figure 4. A DTZ annotation for a single dimension and three zones. The annotations store the graph distance between each node and the closest node in each zone. For example, the node in the extreme lower-left of the graph stores values of [0, 2, 4], corresponding to the distances to the white, black, and gray colored zones. The distance between this node and the node in the extreme lower-right is $4 + 5 = 9$.

For the Girvan-Newman algorithm, we use NSI-guided best-first search to approximate edge betweenness centrality, a technique demonstrated by Rattigan et al. (2006). This method estimates betweenness by sampling pairs of nodes and performing searches between them. As seen in Figure 6, to determine the edge with the highest betweenness score (rather than a rank order of all the edges), it is only necessary to sample a very small number of pairs. Additionally, we can use the index constructed on the entire graph to perform searches throughout the clustering process rather than building indices on the individual connected components. Whereas the Girvan-Newman algorithm requires $O(|E|^2|V|)$, our NSI-based version takes only $O(|E|)$.

4. Results

4.1. Synthetic data sets

We evaluated the NSI-based version of graph k-medoids in the same manner as described in Section 2.2. Since the distances provided by the NSI are approximate, one may expect the clustering performance of graph k-medoids to suffer. As shown in Figure 7, however, the NSI-based clustering actually *outperforms* the exact method, often by a sizable margin. The effect is most pronounced in graphs with inter-cluster linkage between 0.2 and 0.6. At higher levels, the effect disappears, as these graphs have so few intra-cluster links that the link structure no longer encapsulates communities. Even after applying the modal reassignment post-processing, the DTZ clustering still consistently outperforms graph k-medoids with exact distances.

To understand the performance benefit, we must ex-

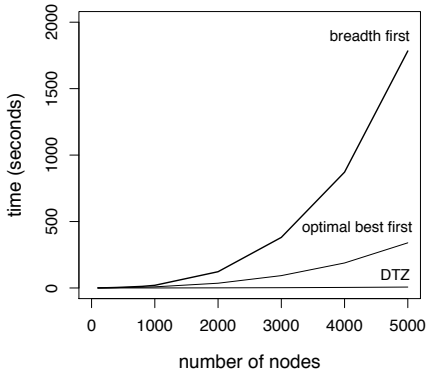

Figure 5. Run time as a function of graph size for the k-medoids algorithm for three methods of calculating geodesic distances between nodes. The top line shows bidirectional breadth-first search, which can become intractable for even moderate-size graphs. The middle line shows an optimal best first search, which represents a lower bound on the run time for any search-based method. The lower line shows an NSI-based method (DTZ with 10 dimensions, 20 zones).

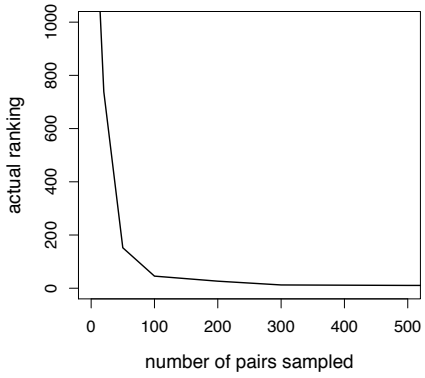

Figure 6. Effectiveness of a sampled betweenness cenrality measure at identifying the top-ranked edge. The average actual rank (out of 10,000) of the estimated top-ranked edge is depicted as a function of the number of pairs sampled. Highly-ranked edges are correctly identified with only a few hundred pairs of nodes, as opposed to the million utilized by the exact measure.

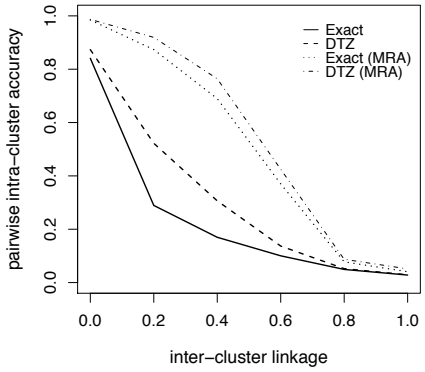

Figure 7. Graph k-medoids performance using exact and DTZ-based distances. The NSI consists of 20 zones and 10 dimensions (larger values of z and d slightly increases performance).

amine the DTZ index more closely. The annotations contain the distances between each node and other sets of nodes (zones). To estimate the distance between a pair of nodes, the distance measure sums the distance from one node to the other node's zone and vice verse. For example, consider the distance calculation from node a_1 to node b_1, given an index with a single dimension. In this example, if a_1's zone (A) consists of nodes$\{a_1, a_2, a_3\}$, and b_1's zone (B) has $\{b_1, b_2\}$, we have:

$$\begin{aligned}\text{dist}_{\text{DTZ}}(a_1, b_1) &= \text{dist}(a_1, B) + \text{dist}(b_1, A) \\ &= \min[\text{dist}(a_1, b_1), \text{dist}(a_1, b_2)] + \\ &\min[\text{dist}(b_1, a_1), \text{dist}(b_1, a_2), \text{dist}(b_1, a_3)]\end{aligned}$$

Thus, for a single dimension, DTZ distance is guaranteed to underestimate actual graph distance. It follows that for multiple dimensions, $\text{dist}_{\text{DTZ}}(a_1, b_1) \leq 2d \times \text{dist}(a_1, b_1)$. If there are many intra-cluster paths between a given node and its true medoid, there is a greater probability of having a smaller estimated distance to the medoid, resulting in a correct assignment. Lowering the value of z (the number of random partitions) effectively increases the effect of underestimating graph distance since each zone will contain more nodes (i.e., the number of terms in the minimization above increases). There is a tradeoff, though: Too low a z and nodes will appear to be closer to many medoids; too high a z, and DTZ distance estimates will approach exact graph distance. Figure 8 depicts the performance effects of choosing different settings for these parameters on graphs with 100 clusters of size 10 and intercluster linkage of 0.2. Similar experiments on graphs with larger clusters produced less easily interpreted results, including non-monotonic performance increases as values of z and d increase. The effects of NSI parameters on performance remains an important area for future work. Furthermore, if the graph contains "noisy" edges, the estimates provided by the NSI can serve to smooth over anomalous links, producing more stable clusterings.

It is important to note that the DTZ performance improvement is not due to random estimation error alone. Figure 9 illustrates this fact by depicting the intra-cluster accuracy of a version of graph k-medoids that utilizes a distance function whose outputs have been perturbed with varying levels of Gaussian noise.

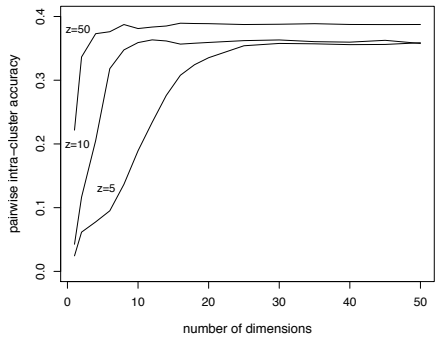

Figure 8. Parameter exploration of the k-medoids algorithm

As the level of noise increases, performance worsens, even if the perturbations are restricted to underestimate graph distance (reflecting the approximate nature of DTZ). Clearly, the performance-enhancing errors of DTZ's distance measures are correlated with graph structure rather than entirely stochastic.

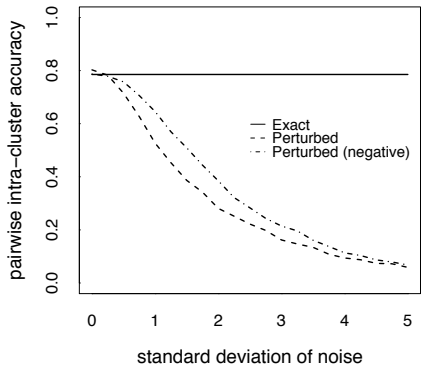

Figure 9. The effect of random distance perturbations on graph k-medoids clustering performance. Distance estimation error that is generated randomly greatly decreases accuracy. This contrasts with the accuracy-enhancing error associated with DTZ depicted in Figure 7.

As with graph k-medoids, the NSI-based version of the Girvan-Newman algorithm performs extremely well. Figure 10 depicts pairwise intra-cluster accuracy over a range of graph types. The accuracy value stays above 0.8 for graphs with inter-cluster linkage up to 0.4. The accuracy drops off as the structural separation between clusters starts to lessen. Like graph k-medoids, the inter-cluster accuracy remains over 0.95 across the range. Here MRA increases the performance somewhat, though not as dramatically as before. While this method returns more accurate clusterings than graph k-medoids, it comes at a significant cost, as on average its run time was several orders of magnitude larger for moderate-size graphs.

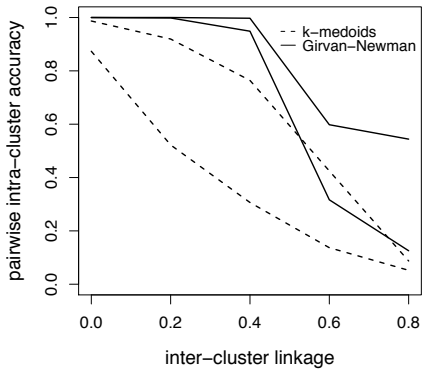

Figure 10. Girvan-Newman clustering performance with a DTZ NSI of 50 zones, 10 dimensions. MRA improves the accuracy slightly and is most pronounced at higher levels of inter-cluster linkage.

4.2. Real data sets

We tested k-medoids on two real relational data sets (the slower Girvan-Newman algorithm was not run on these datasets for perforamance reasons). The first was a network of over 16,000 actors drawn from the Internet Movie Database (http://www.imdb.com). Actors are connected by links when they have acted together in at least two films or television shows produced between 1970 and 2000. Using graph k-medoids with a DTZ NSI, the algorithm consistently coverged in less than five iterations for $k = 300$ clusters. Table 1 below shows three examples of well-formed actor communities. The leftmost group consists of actors from the various *Star Trek* movies and television series. The middle group is composed of players from the television series *Saturday Night Live*, while the third group represents actors from various Kevin Smith films.

Table 1. Three example clusters of actors discovered by k-medoids: the Kevin Smith, *Star Trek*, and *Saturday Night Live* clusters. These examples rank 19th, 40th, and 54th, repectively, out of 300 when ranked by inter-cluster linkage. The actors in boldface were selected to be the cluster medoids.

Frakes, Jonathan	Lovitz, Jon	Adams, Joey Lauren
Shatner, William	**Sandler, Adam**	Affleck, Ben
Warner, David	Schneider, Rob	Ewell, Dwight
Dorn, Michael	Farley, Chris	Lee, Jason
Spiner, Brent	Covert, Allen	Damon, Matt
Stewart, Patrick	Macdonald, Norm	Suplee, Ethan
Schuck, John	Farley, John	Smith, Kevin
Nimoy, Leonard	Nealon, Kevin	Mewes, Jason
Doohan, James	Dante, Peter	Flanagan, Walter
Koenig, Walter	Clark, Blake	Rock, Chris
Overton, Rick	Smigel, Robert	OHalloran, Brian
Takei, George	Titone, Jackie	**Mosier, Scott**

The second domain we examined was a citation net-

work generated by the Cora project (McCallum et al., 1999). In the Cora dataset, the nodes of the graph represent over 30,000 scientific papers, connected by over 130,000 links. Two papers are connected by a link if one cites the other. In addition, each paper is assigned to one of 71 topics learned from the text of their titles and abstracts. Using graph k-medoids, we attempted to recover the topic groups by clustering the papers using their link structure. A visual representation of the relationships between topic groups and clusters can be seen in Figure 11. To evaluate our clustering, we ran Pearson's Chi-squared test on the predicted clusters and their associated topic distribution.

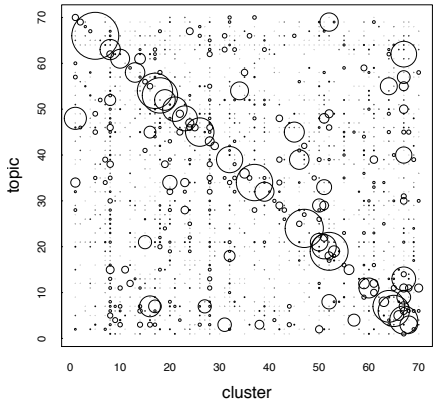

Figure 11. Visual representation of the relationship between cora topic groups and the predicted clusters procuced by graph k-medoids. The dots are sized relative to the number of papers for each topic in each cluster. The lack of rows or columns with more than one dominant circle indicates highly correlated relationships between topics and clusters, and 31% of the mass of the total topic distribution lies on the diagonal.

5. Related work

Graph clustering has been studied in numerous scientific communities, ranging from physics (Newman, 2004b) to social networking analysis (Freeman, 1979) to computer science (Flake et al., 2004). Newman (2004) provides an excellent overview of some of the most well known techniques for clustering graphs. Most approaches fall into one of several algorithmic categories, detailed below.

The procedures in the first set follow the agglomerative clustering paradigm. For these algorithms, each node starts off in its own cluster, which are then combind in a principled manner until the desired number of clusters is achieved, or the threshold for some graph measurement is reached. Among these methods are Newman's algorithm (Newman, 2004b). King presents a cost-based algorithm for assembling a clustering via agglomerative (or divisive) movement of nodes between clusters (King, 2004).

The second main group of graph clustering algorithms relies on edge removal to achieve a clustering among the connected components of the graph. The Girvan-Newman technique (and our approximation of it) fall into this category. Van Dongen (2004) presents an edge removal algorithm that is based on network flow simulation. Flake, Tarjan, and Tsioutsiouliklis (2004) present an algorithm based on finding the minimum cut tree of a graph. While these methods yield encouraging results, they are not scalable to large graphs.

There is also a great deal of literature on spectral methods, such as those outlined by Ng (Ng et al.,). For these techniques, eigenvectors are calculated for the Laplacian of the graph adjacency matrix. The nodes of the graph are then clustered by performing traditional data clustering on the eigenvectors. Related are methods that project link structure into some vector space (rather than using eigenvectors); examples include Huang, Dhillon, and Handcock (Huang & Lai, 2006) (Dhillon et al., 2004) (Handcock et al., 2005). While these methods produce accurate results, they can be computationally infeasible for large graphs.

As far as we know, the k-medoids algorithm has not been applied in a graph clustering context. Hlaoui & Wang and Schenker et al. present a version of k-means for graphs; however, in their domain the data consist of IID subgraphs that must be clustered together, rather than individual nodes in the same graph (Hlaoui & Shengrui, 2004), (Schenker et al., 2003).

6. Conclusions and future work

In spite of their simplicity, the graph k-medoids and Girvan-Newman algorithms are surprisingly effective at clustering graphs. Unfortunately, their computational complexities are prohibitively large for even moderately sized graphs. We have shown how a network structure index can be utilized within these algorithmic frameworks to overcome these limitations, achieving equal or better performance results.

There are several future research directions for the current work on NSI-based graph clustering methods. While we found the DTZ NSI to be effective at finding clusters, there is certainly room for improvement in terms of the time and space required to calculate and store the index. Additionally, an NSI for weighted graphs could perhaps improve the clustering performance on highly-connected domains such as the IMDb

actor graph. The NSIs utilized by graph k-medoids are multipurpose; that is, the same index can be used to effectively estimate centrality measures. It is possible that a more specialized type of index, tailored to the clustering task, could produce better results.

In addition to a defined link structure, many relational data sets contain attribute information on the links and objects. Another direction for this work is to adapt it to domains with attributes, and devise a principled way to incorporate them into graph k-medoids. Finally, our work entirely ignored the issue of choosing the proper number of clusters, when applying the algorithms. It may be possible to use NSI-based measures to choose k in a principled manner.

Acknowledgments

This effort is supported by The National Science Foundation, the Central Intelligence Agency, and the National Security Agency under contract number IIS-0326249. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of the U.S. Government or any agency thereof.

References

Brandes, U., Gaertler, M., & Wagner, D. (2003). Experiments on graph clustering algorithms. *Proceedings of the 11th Annual European Symposium on Algorithms (ESA03)*, *2832*, 568–579.

Coppersmith, D., & Winograd, S. (1987). Matrix multiplication via arithmetic progressions. *Proceedings of the Nineteenth Annual ACM Conference on Theory of Computing*, 1–6.

Delling, D., Gaertler, M., & Wagner, D. (2006). Generating significant graph clusterings. *Proceedings of the European Conference of Complex Systems ECCS '06.*

Dhillon, I., Guan, Y., & Kulis, B. (2004). Kernel k-means: spectral clustering and normalized cuts. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, 551–556.

Flake, G., Tarjan, R., & Tsioutsiouliklis, K. (2004). Graph clustering and minimum cut trees. *Internet Mathematics*, *1*, 385–408.

Freeman, L. (1979). Centrality in social networks. *Social Networks*, *1*, 215–239.

Girvan, M., & Newman, M. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, *99*, 7821–7826.

Handcock, M., Raftery, A., & Tantrum, J. (2005). *Model-based clustering for social networks* (Technical Report). Working Paper 46, Center for Statistics and the Social Sciences, University of Washington.

Hlaoui, A., & Shengrui, W. (2004). A direct approach to graph clustering. *Proceedings of the 2nd IASTED International Conference on Neural Networks and Computational Intelligence, NCI 2004, Grindelwald, Switzerland.*

Huang, X., & Lai, W. (2006). Clustering graphs for visualization via node similarities. *Journal of Visual Languages and Computing*, *17*, 225–253.

Kaufman, L., & Rousseeuw, P. (1990). Finding groups in data. an introduction to cluster analysis. *Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics, New York: Wiley, 1990.*

King, A. (2004). Graph Clustering with Restricted Neighbourhood Search. Master's thesis, University of Toronto.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, *1*, 281–297.

McCallum, A., Nigam, K., Rennie, J., & Seymore, K. (1999). A machine learning approach to building domain-specific search engines. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 662–667.

Newman, M. (2004a). Detecting community structure in networks. *The European Physical Journal B-Condensed Matter*, *38*, 321–330.

Newman, M. (2004b). Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, *69.*

Ng, A., Jordan, M., & Weiss, Y. On spectral clustering: Analysis and an algorithm.

Rand, W. (1971). Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, *66*, 846–850.

Rattigan, M., Maier, M., & Jensen, D. (2006). Using structure indices for efficient approximation of network properties. *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 357–366.

Schenker, A., Last, M., Bunke, H., & Kandel, A. (2003). Graph representations for web document clustering. *Proc. 1st Iberian Conf. on Pattern Recognition and Image Analysis.*