Name Disambiguation Using Multi-Level Multi-Resolution (MLMR) Graph Partitioning

Ingyu Lee[1] **and Byung-Won On**[2]
[1]Sorrell College of Business, Troy University, Troy, AL., USA
[2]Department of Computer Science, University of British Columbia, Vancouver, BC., Canada

Abstract— *When we are looking for information about a certain person, we type the name on search engines. Then, search engines return many related web pages which includes the name string. However, some web pages are directly related with the one we are looking for but other web pages include the same name string in the contents by coincidence. This is called a name disambiguation problem. With the increasing number of web pages, finding related web pages becomes more important. In this paper, we propose a multi-level and multi-resolution graph partitioning algorithm (MLMR) to solve a name disambiguation problem. Our experiments show that MLMR improved the performance by approximately 20% compared to Metis and 10% compared to KMeans. We believe that search engines will provide a better service by combining our MLMR algorithm.*

Keywords: data mining, clustering, graph partitioning, Multi-level algorithm

1. Introduction

Internet allows us to search for information about people and to work with people whom we never met in real world. When we are looking for information about a specific person, we type the name string in search engines. Then, search engines return many related web pages from World Wide Web. Some web pages are directly related with whom we are looking for but other web pages include the same name string by coincidence. Current search engines do not distinguish the differences between web pages. This is called a name disambiguation problem. With the increasing number of web pages, finding directly related web pages becomes more and more important. For example, if we are looking for some information about *Tom Mitchell* from Carnegie Mellon University, we type *Tom Mitchell* in Google search engine. Then, Google returns links of web pages including the name string in their contents. Within the first 100 Google returning web pages, there are 37 different *Tom Mitchell*s [3]. Among the first 100 web pages only 57 documents hold the information about *Tom Mitchell* from Carnegie Mellon University and all the other documents including information about other *Tom Mitchell*s. Figure 1 shows the search results in Google. The first link shows information about *Tom Mitchell* from Carnegie Mellon University but all the other links are information about other *Tom Mitchell*s.

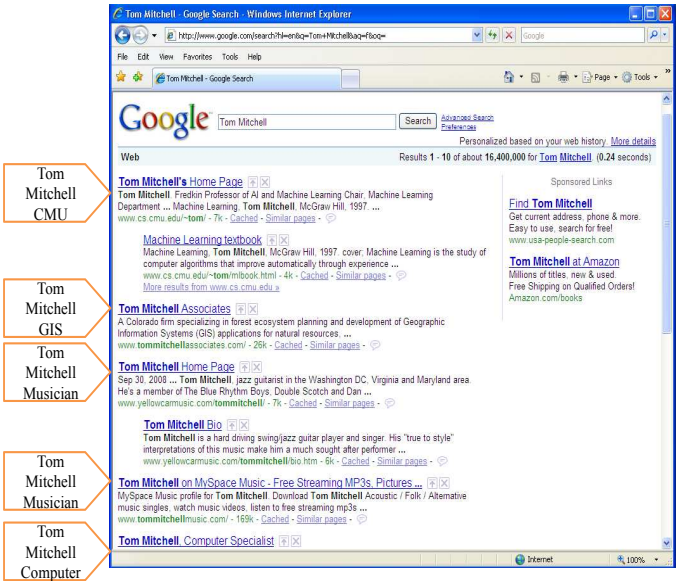

Fig. 1: Google Search Results of *Tom Mitchell*.

Many researches have been done to provide solutions of a name disambiguation problem [1], [2], [17], [18]. One way to solve a name disambiguation problem is clustering the search results into groups of related web pages using algorithms such as K-means, spectral, and graph partitioning. Let's assume we have n web documents which includes k different personal information. Then the name disambiguation problem is defined as

$$\min \sum_{i=1,\ldots,k} \sum_{d_j \in C_i} (d_j - m_i)^2 \qquad (1)$$

where d_j is a document $j = 1, \ldots, n$ and m_i is a centroid of cluster C_i for $i = 1, \ldots, k$. Kmeans [11], [16] is a simple and most popular algorithm in clustering. However, the convergence of Kmeans depends on the initial centroid points. In other hand, spectral clustering methods [11], [19] show good performance with a little bit of operational overhead. However, when dimensions become large, Kmeans and spectral algorithms are infeasible to apply. For larger data set, multi-level graph partitioning algorithms have been used to provide scalable solutions [14], [15]. A name disambiguation problem has variant sizes of clusters and the cluster size distribution is extremely skewed. Therefore, traditional

graph partitioning algorithms do not properly cluster data sets for a name disambiguation problem.

In this paper, we propose a multi-level and multi-resolution graph partitioning algorithm. This algorithm especially useful when we partition extremely skewed graphs which have variant sizes of clusters. Our algorithm combines a multi-level with a multi-resolution spectral clustering algorithm. Multi-level property provides scalable algorithms even with huge number of web pages and multi-resolution allows us to cluster extremely skewed name data sets.

This paper consists of the followings. In Section 2, we provide some background of graph partitioning algorithms especially spectral bisection and multi-level graph partitioning. Our multi-level and multi-resolution algorithm (MLMR) is described in Section 3. Experimental results with web naming data sets are presented in Section 4. Concluding remarks and future plans are followed in Section 5.

2. Graph Partitioning

A graph $G = (V, E)$ is defined in terms of a set of vertices V and a set of edges E. Graph partitioning is defined as a function to assign each vertices to a subgraph G_i, where $i = 1, \ldots, m$ and m is the number of clusters. A goal of graph partitioning is to find subsets $\{G_1, G_2, \ldots, G_m\}$ which minimizes the connection between subgraphs

$$\min \sum_{i,j \in (1,\ldots,m)} \frac{Cut(G_i, G_j)}{|G_i| + |G_j|} \tag{2}$$

and, at the same time, maximizes the edges within the subgraphs such as

$$\max \sum_{i,j \in (1,\ldots,m)} \frac{Edges(G_i) + Edges(G_j)}{|G_i| + |G_j|} \tag{3}$$

where $Cut(G_i, G_j)$ is the number of edges connecting two subgraphs G_i and G_j, $Edges(G_i)$ is the number of edges in the subgraph G_i, and $|G_i|$ is the number of vertices in the subgraph.

Spectral clustering is based on global connectivity unlike other graph partitioning algorithms which depend on local information, as shown in [19]. To apply spectral clustering, we first generate a similarity graph from original graph G. In the similarity graph, $G_s(i, j)$ has 1 if $G(i, j) \neq 0$ and the diagonal element $G_s(i, i)$ has the number nonzero elements in that row i. After constructing a similarity graph G_s, we find the second smallest eigenvalue and a corresponding eigenvector which is known as a fiedler vector. Finally, we divide the graph into two subgraphs using the sign of the components of the fiedler vector [9].

Other approach that has greatly accelerated the partitioning of graphs is the use of multi-level techniques. In multi-level, we are constructing a hierarchy of approximations to the original graph so that a coarse graph can be quickly partitioned. Then, the solution is progressively refined until an original graph is reached. These techniques are very similar to a multi-grid method [4] for solving numerical problems. Especially, with the increasing number of web pages, multi-level methods provide scalable algorithms as shown in [14], [15].

A multi-level approach has three phases: coarsening phase, partitioning phase and uncoarsening phase. Each of the phases are described in the followings.

1) Coarsening Phase. The graph G_0 is transformed into a sequence of smaller graphs $G_1, G_2, \ldots, G_m$ such that $|V_0| > |V_1| > |V_2| > \cdots > |V_m|$. At each level, we applied an approximation algorithm to convert from G_i to G_{i+1} until we reached a reasonably small graph G_m.
2) Partitioning Phase. A 2-way graph partition P_m of the graph $G_m = (V_m, E_m)$ is computed that partitions V_m into two parts, each containing approximately half the vertices of G_m. After the graph G_m becomes small enough, we apply Kernighan and Lin [7] or Fiduccia Mattheyses [8] algorithm to refine the partition.
3) Uncoarsening Phase. The partitioning P_m of G_m is projected back to G_0 by going through intermediate partitions $P_{m-1}, P_{m-2}, \ldots, P_1, P_0$. At each level, we applied uncoarsening algorithm to expand a graph G_{i+1} into a graph G_i.

3. Multi-Level and Multi-Resolution (MLMR) Graph Partition Algorithm

In this section, we describe the characteristics of name data sets and propose an algorithm to solve the problem. We used name data set from Bekkerman [3] to analyze the characteristics of the data. It has twelve personnel names and each name set has different number of categories. The cluster size distribution is also extremely skewed and biased. For example, *Cheyer* has 97 documents with 2 classes but only one document belongs to one class and the other class has 96 documents. *Kaebling* shows the similar characteristic as shown in Table 1. In another case, *Cohen* has 10 clusters but one cluster dominates the data set with 67 documents and the other 7 clusters each has only one document. Figure 2 summarizes the distribution of cluster sizes for each name data sets. Each color shows different classes and the sizes of color bar shows the number of documents belong to the cluster.

Therefore, conventional graph partitioning algorithms, which are based on the assumption that cluster sizes are similar or evenly distributed, do not work properly on name data sets. To overcome the latter, we propose a multi-level and multi-resolution (MLMR) graph partitioning algorithm. MLMR algorithm is based on multi-level graph partitioning but allow multi-resolutions with variant levels. At each step, the algorithm decides to go further to next level only if the partitioning produces more inter cluster edges than

Name	Pages	Clusters	Max
Adam Cheyer	97	2	96
William Cohen	88	10	67
Steve Hardt	81	6	64
David Israel	92	19	38
Leslie Kaebling	89	2	88
Bill Mark	94	8	54
Andrew McCallum	94	16	54
Tom Mitchell	92	37	15
David Mulford	94	13	56
Andrew Ng	87	31	32
Fernando Pereira	88	19	32
Lynn Voss	89	26	25
Total	**1085**	**187**	

Table 1: Dataset Statistics. Each name has different number of documents and different number of categories.

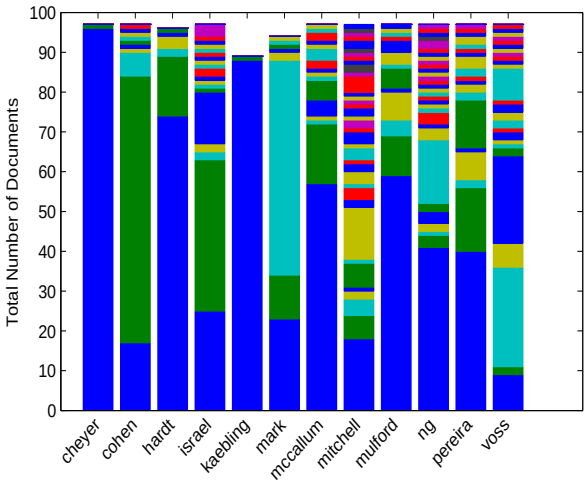

Fig. 2: Problem Statistics. Each color represent different clusters.

intra cluster edges. Otherwise, MLMR algorithm stops at the current level. In the end, MLMR algorithm generates different levels for each branch and various resolutions for each leaf node.

MLMR graph partitioning algorithm has two phases: graph partitioning and graph merging. In graph partitioning phase, we applied spectral bisection algorithm recursively to divide a given graph into two subgraphs until subgraphs become a clique or reach to a condition where subgraphs have more inter subgraph edges than intra subgraph edges. After dividing graph into small subgraphs, we merge subgraphs together if two subgraphs have the biggest normalized cuts. We repeatly merge subgraphs until the number of subgraphs equal to the given number of clusters. Figure 3 shows the outline of the partition and merging algorithm.

The details of MLMR algorithm is described in Figure 4

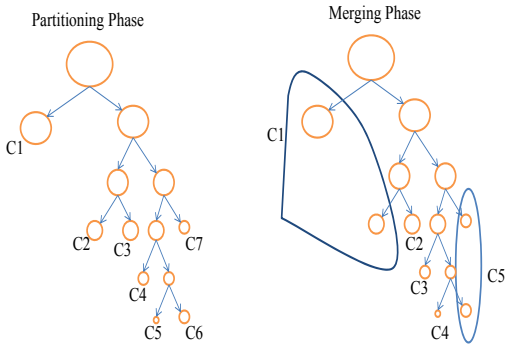

Fig. 3: MLMR graph partitioning.

and Figure 5. After constructing a graph G, we used a spectral bisection algorithm to partition the graph into G_1 and G_2. Once we partition the graph, we compute normalized cut using two subgraphs G_1 and G_2. If the normalized cut is bigger than a threshold value, then we keep the original graph G and stop the algorithm on that branch. Otherwise, we keep the subgraphs and applied the same algorithm recursively on subgraphs G_1 and G_2. This partitioning phase generates different levels for each branch, and the resolutions of each branch is also different.

Once we have gone through the partitioning phase, we applied a merging phase on the partitioned subgraphs $G_1, G_2, \cdots, G_m$. The number of subgraphs m depends on the resolution we choose during the partitioning phase. During the merging phase, we construct a pairwise normalized cut table as shown in Table 6. Then, we merge two subgraphs which have the largest normalized cut. We repeatedly merge subgraphs until the number of subgraphs m reaches the given number of clusters.

At each level, we compute an eigenvalue and a corresponding eigenvector for spectral partitioning. Assume we have N nodes in our initial graph. Then, for level i, we have 2^{i-1} subgraphs with $N/2^{i-1}$ node elements. If we assume the computation of eigenvector takes N^3 for N elements [12], [10], the total computation requires

$$T = N^3 + 2 * \left(\frac{N}{2}\right)^3 + 4 * \left(\frac{N}{4}\right)^3 + \cdots + N * \left(\frac{N}{N}\right)^3 \quad (5)$$

which is simplified as

$$T = \frac{4}{3} N^2 (N - 1) \quad (6)$$

in an ideal case where a graph G is divided into two approximately equal size subgraphs G_1 and G_2.

- **Graph Partitioning Phase:**
  1) We build a graph $G = (V, E)$ from a term-document matrix A by $A^T A$. Since $A^T A$ is a completely dense graph, we do not explicitly construct $A^T A$. Rather, we keep only values which are larger than a $threshold$ value. Intuitively, this process filters out edges for remotely related documents and keeps only edges for strongly related documents.
  2) Then, we divide the graph G into two subgraphs G_i and G_j using spectral bisection and compute normalized cuts between two subgraphs G_i and G_j. Normalized cuts between subgraphs G_i and G_j is defined as
  $$NormCut_{i,j} = \frac{Cut(G_i, G_j)}{Edge(G_i) + Edge(G_j)} \quad (4)$$
  where $Cut(G_i, G_j)$ is the number of edges connecting two subgraphs and $Edge(G_i)$ is the number of edges within subgraph G_i.
  3) If the normalized cut $NormCut_{i,j}$ is small, then we divide the graph G into G_i and G_j, and continuously applying the partitioning algorithm. Otherwise, we keep the original graph G and stop the partition.

Fig. 4: The Graph Partitioning Algorithm

- **Graph Merging Phase:**
  1) We compute the intra subgraph edges for each subgraph G_i and inter subgraphs edges for each pair of subgraphs G_i and G_j. The pairwise computation cost depends on the subgraph resolution during the partitioning phase.
  2) Subgraph G_i and G_j will be merged into a graph G if the normalized cut between two subgraphs G_i and G_j has the maximum value in $\{G_1, G_2, \ldots, G_n\}$ where n is the number of subgraphs.
  3) If the number of subgraphs n is smaller than the given number of cluster k, then we repeatedly apply the merging algorithm until the number of subgraphs reaches to the given number of clusters.

Fig. 5: The Graph Merging Algorithm

	G_1	G_2	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	G_n
G_1	•	m_1	m_2	m_3	m_4	m_5	m_6	m_7
G_2		•	m_8	m_9	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\cdots$			•	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\cdots$				•	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\cdots$					•	$\cdots$	$\cdots$	$\cdots$
$\cdots$						•	$\cdots$	$\cdots$
$\cdots$							•	$\cdots$
G_n								•

Fig. 6: Normalized cut merge table between subgraphs.

During the merging phase, we compute a ratio between the number of edges in the graph G_i and G_j, and the number of edges connecting graph G_i and G_j for each pair $i, j = 1, \ldots, n$ where n is the number of subgraphs in the partitioning phase. It requires $\frac{1}{2}n^2$ computations until the number of clusters n reaches to k which is a given number of clusters. Therefore, the total computation will be

$$\frac{1}{2}(n-k) * n^2 \quad (7)$$

where n depends on the resolution in the partitioning phase and k is the number of clusters in the solution. Our partition resolution n is very close to the solution k, then merging phase cost becomes minimal.

4. Empirical Results

To mesure the performance, we used name set data from Bekkerman [3]. First, we constructed a term document matrix A with normalization and stemming options using TMG [21] software. We also deleted commonly used terms from the terminology dictionary. Term frequency and document inverse frequency are used for local and global term weighting, respectively. Then, we constructed a document-document graph matrix $G = A^T A$ by keeping only values which are bigger than a *threshold*. We used .3 for our threshold value. Finally, we applied Metis graph partitioning and MLMR algorithm to cluster the documents. We measured precision, recall, and Fmeasure. Precision is the ratio between the number of correctly predicted documents and the number of predicted documents. Recall is the ratio between the number of correctly predicted documents and the number of documents belongs to that category from a solution set. Fmeasure is an arithmetic average of precision and recall.

We also measured RAND index which has been used in statistics community to compare two clustering results [20]. Given a set of n documents $D = \{d_1, \ldots, d_n\}$, we wish to compare two clusters: C and S. The resulting cluster of our algorithm is defined as $C = \{C_1, \ldots, C_k\}$ and manually derived solution set is defined as cluster $S = \{S_1, \ldots, S_k\}$. Then, RAND index is defined as

$$RAND = \frac{a+b}{a+b+c+d} \quad (8)$$

where a is the number of pairs of elements in D that are in the same set in C and the same set in S, b is the number of pairs of elements in D that are in different sets in C and in different sets in S, c is the number of pairs of elements in D that are in the same set in C and in different sets in S, and d is the number of pairs of elements in D that are in different sets in C and in different sets in S. Intuitively, $a + b$ is the

Name	Metis				MLMR			
	Pr	Re	Fm	Rd	Pr	Re	Fm	Rd
Adam Cheyer	.99	.5	.66	.49	.99	.5	.66	.96
William Cohen	.76	.18	.29	.44	.98	.25	.4	.61
Steve Hardt	.78	.16	.27	.42	.88	.25	.39	.61
David Israel	.58	.16	.26	.75	.94	.33	.49	.47
Leslie Kaebling	.98	.5	.66	.49	.99	.5	.66	.95
Bill Mark	.74	.27	.4	.62	.94	.39	.55	.45
Andrew McCallum	.72	.19	.31	.65	.98	.62	.76	.62
Tom Mitchell	.59	.32	.42	.93	.96	.47	.63	.72
David Mulford	.69	.25	.37	.64	.93	.42	.58	.6
Andrew Ng	.68	.33	.45	.82	.95	.55	.69	.67
Fernando Pereira	.57	.16	.26	.8	.94	.37	.53	.46
Lynn Voss	.68	.32	.43	.84	.95	.39	.56	.76
Average	**.73**	**.27**	**.39**	**.65**	**.95**	**.42**	**.57**	**.65**

Table 2: Experimental results of Metis, and MLMR on name dataset. Pr: Precision, Re: Recall, Fm: Fmeasure, Rd: Rand Index.

Name	KMeans				MLMR			
	Pr	Re	Fm	Rd	Pr	Re	Fm	Rd
Adam Cheyer	.99	.5	.66	.57	.99	.5	.66	.96
William Cohen	.82	.16	.27	.54	.98	.25	.4	.61
Steve Hardt	.85	.22	.36	.48	.88	.25	.39	.61
David Israel	.76	.22	.34	.72	.94	.33	.49	.47
Leslie Kaebling	.98	.5	.66	.95	.99	.5	.66	.95
Bill Mark	.83	.23	.36	.57	.94	.39	.55	.45
Andrew McCallum	.86	.29	.43	.67	.98	.62	.76	.62
Tom Mitchell	.79	.44	.56	.92	.96	.47	.63	.72
David Mulford	.88	.24	.38	.59	.93	.42	.58	.6
Andrew Ng	.86	.35	.5	.74	.95	.55	.69	.67
Fernando Pereira	.82	.27	.4	.79	.94	.37	.53	.46
Lynn Voss	.88	.32	.47	.74	.95	.39	.56	.76
Average	**.86**	**.31**	**.45**	**.69**	**.95**	**.42**	**.57**	**.65**

Table 3: Experimental results of Kmeans and MLMR on name dataset. Pr: Precision, Re: Recall, Fm: Fmeasure, Rd: Rand Index.

number of agreements between C and S, and $c + d$ is the number of disagreements between C and S. RAND index has a value between 0 and 1 with 0 indicating that two data clusters do not agree on any pair of points and 1 indicating that two data clusters are exactly the same.

Table 2 shows the performance comparison of Metis and MLMR algorithm on 12 different name data sets. MLMR shows *20%* better than Metis in terms of precision, *15%* better in terms of recall, and *18%* better in terms of Fmeasure. For the Rand Index, the Metis shows better numbers when the name data sets are relatively evenly spread out among the data set. Metis is designed to divide a graph into equal sized subgraphs which is important in load balancing for parallel computing. However, in general, name data set is extremely skewed and one cluster in the set dominates the whole data set as described in the previous section.

Table 3 shows the performance comparison between KMeans and MLMR clustering algorithm. Even compared with KMeans, MLMR shows approximately *10%* better performance in terms of precision, recall, and Fmeasure. However, in Rand Index, KMeans shows a slightly better performance than MLMR in average. KMeans shows better Rand Index in *5* data sets and MLMR shows better numbers in *3* data sets. The other data sets show similar Rand Index numbers on both algorithms.

Figure 7 summarizes the performance results with average performance for 12 different name data sets. MLMR algorithm shows approximately 10% better performance compared to the conventional K-means algorithm in terms of precision, recall and Fmeasure with similar number in Rand Index. Compared to Metis, MLMR algorithm shows approximately *20%* better performance in terms of precision. Considering Metis is designed to share the loads in parallel computing environments, the experimental results are predictable. From Rand Index point, KMeans shows the better performance since MLMR is based on approximation.

To understand how the algorithms catch the cluster size

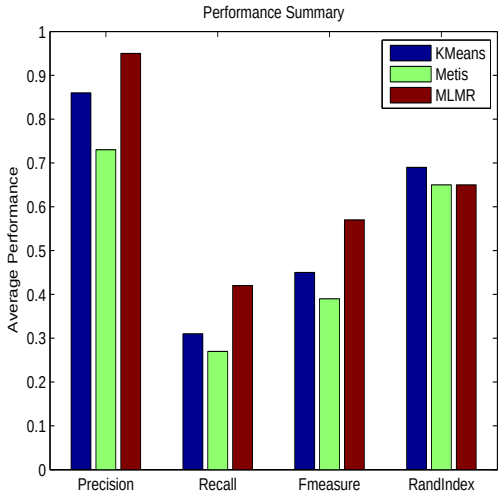

Fig. 7: Performance Summary.

distribution, we analyzed the spectrum of cluster sizes of the three methods. Figure 8 shows the spectrum analysis of clustering results of Kmeans, Metis, and MLMR for 12 name data sets. After clustering name data sets, we sorted clustering results in ascending order in cluster size. The first line shows the spectrum of manually clustered solution set, and Kmeans, Metis and MLMR results are followed in that order. The spectrum shows how close the clustering results to a manually clustered solution set in terms of cluster size. Metis always divides the data set into similar size clusters as shown in the Figure 8. Kmeans shows close spectrum to a manual solution in some data sets such as *Mark*, and *McCallum*. However, in all the other name data sets, MLMR shows closer spectrum to a manual solution. We can conclude that MLMR finds the cluster size distribution better than the other algorithms.

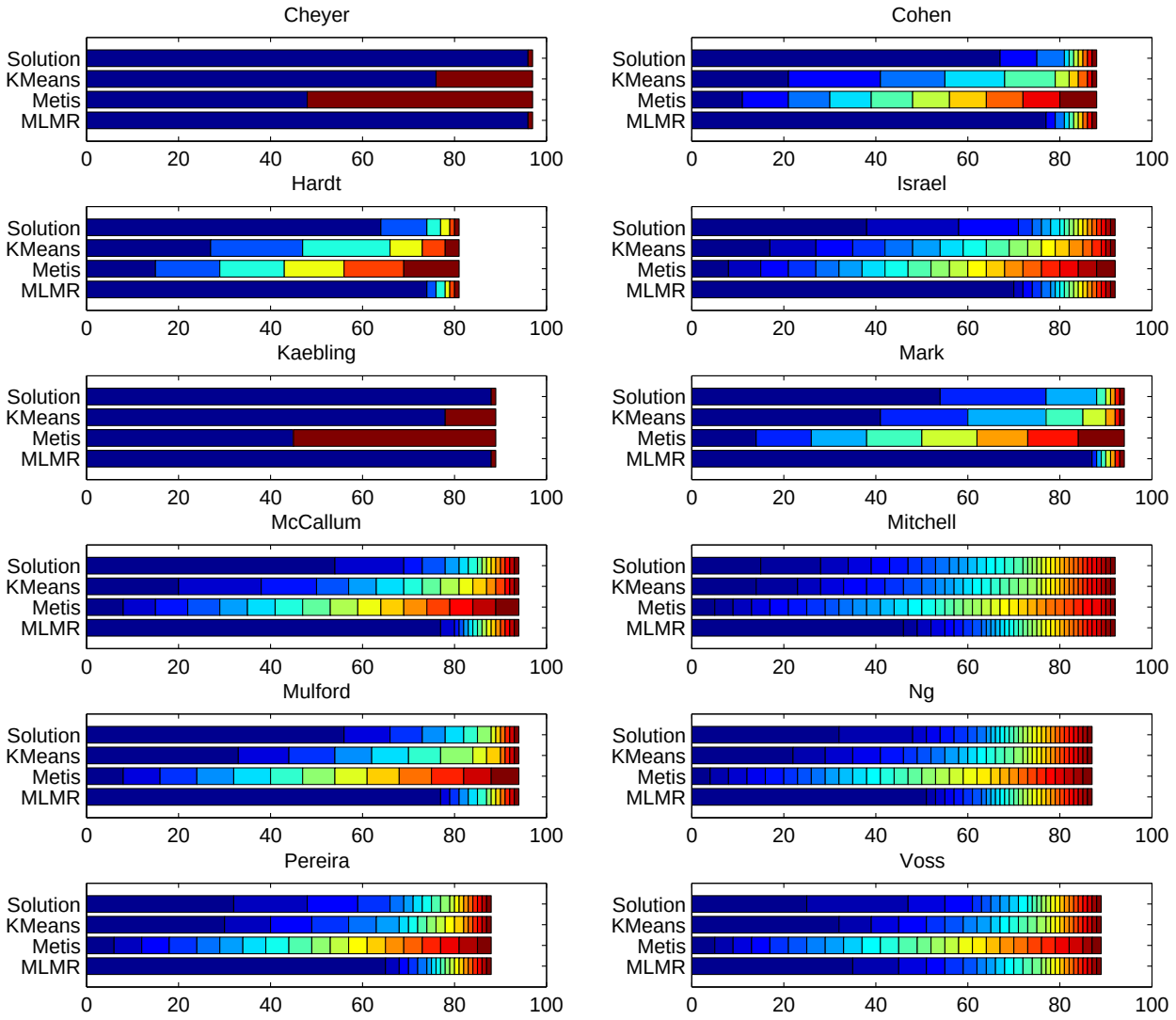

Fig. 8: Spectrum Analysis for Name Data Set.

5. Related Works

Graph partitioning has been studied in parallel computing community. Main goal is to divide the workloads among computing nodes and to reduce the communications between computing nodes. To support these goals, they focus on dividing a graph into similar sizes of subgraphs to balance the workload between computing nodes with minimal communications. Metis [14], [15] and Chaco [13] are software packages popularly used to partition graph into balanced subgraphs. However, a name disambiguation problem consists of extremely skewed and biased classes.

In data mining community, other methods to divide into unbalanced subgraphs has been studied for web and text documents clustering. Graclus [6] has been proposed to divide graph into unbalanced way to fit well in skewed clustering. Experimental results showed the better performance than Metis for some data sets. eigenCluster [5] has been proposed to provide a solution integrating clustering algorithm with a search engine. They focused on unsupervised learning and used a dynamic program to merge clusters after partitioning. As far as authors aware, MLMR algorithm is the only one that allows variant levels and resolutions during partitioning

phase.

Bekkerman et. al. proposed two algorithms to disambiguating web appearances of people in a social network in their paper [2]. One is based on link structure of web pages, another using multi-way distributional clustering method. Minkov et. al. used a lazy graph walk algorithm to disambiguating names in email documents in their paper [17]. They provided framework for email data, where content, social networks and a timeline to integrated in a structured graph. Banerjee et. al. proposed a multi-way clustering on relation graphs in [1]. Different types of entities are simultaneously clustered based not only on their intrinsic attribute values, but also on the multiple relations between entities. On and Lee used multi-level graph partitioning methods to provide a scalable name disambiguation solution in their paper [18].

6. Concluding Remarks

In this paper, we analyzed the characteristics of a name disambiguation problem and we also provided a multi-level and multi-resolution (MLMR) graph partitioning algorithm for extremely skewed and biased class clustering. Experiments show that MLMR algorithm improves the performance by *10%* in terms of precision, recall and Fmeasure with similar Rand index compared to KMeans. We also showed the spectral analysis of clustering results of three methods and compared with a manual solution. Metis divides the graph into evenly among clusters, and Kmeans shows better spectrum in detecting skewed classes for *2* data sets. However, MLMR shows the better spectrum results in other *10* data sets.

Current version of MLMR algorithm is based on supervised learning. However, in real web searching environment, the number of clusters is unknown in advance. We are planning to develop an unsupervised algorithm which assumed the number of clusters based on spectrum analysis. After clustering documents, how to display using graphical user interface and how to rank the clustering results are issues to solve before combining with search engines.

References

[1] A. Banerjee, S. Basu, and S. Merugu, "Multi-way clustering on relation graphs," in *Proceedings of SIAM Data Mining 2007*, 2007.

[2] R. Bekkerman and A. McCallum, "Disambiguating web appearances of people in a social network," in *Proceedings of International World Wide Web Conference Committee*, 2005.

[3] R. Bekkerman, "Name data set," http://www.cs.umass.edu/ ronb.

[4] W. L. Briggs, V. E. Henson, and S. F. McCormick, *Multgrid Tutorial*, 2nd ed. Philadelphia, PA: SIAM, 2000.

[5] D. Cheng, R. Kannan, S. Vempala, and G. Wang, "A divide-and-merge methodology for clustering," in *ACM Transactions on Database Systems*, 2005.

[6] I. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors: A multilevel approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29:11, pp. 1944–1957, 2007.

[7] A. Dunlup and B. Kernighan, "A procedure for placement of standard-cell vlsi circuits," *IEEE Tans. CAD*, pp. 92–98, 1985.

[8] C. Fiduccia and R. Mattheyses, "A linear time heuristic for improving network partitions," in *Proc. 19th IEEE Design Automation Conference, IEEE*, 1982.

[9] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Math Journal*, vol. 23, pp. 298–305, 1973.

[10] G. Golub and C. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: Johns Hopkins University Press, 1996.

[11] J. Han, M. Kamber, and A. Tung, *Spatial clustering methods in data mining: A survey. In Geographic Data Mining and Knowledge Discovery*. Taylor and Francis, 2001.

[12] M. Heath, *Scientific Computing: An introductory survey*. Prentice Hall, 2002.

[13] B. Hendrickson and R. Leland, "The chaco user's guide: Version 2.0," Sandia, Tech. Rep., 1994.

[14] G. Karypis and V. Kumar, "Parmetis: Parallel graph partitioning and sparse matrix ordering library," Department of Computer Science, University of Minnesota, Tech. Rep. TR 97-060, 1997.

[15] ——, "A parallel algorithm for multilevel graph partitioning and sparse matrix ordering," *Journal of Parallel and Distributed Computing*, vol. 48, no. 1, pp. 71–95, 1998.

[16] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Sympositum on Mathematical Statistics and Probability*, 1967.

[17] E. Monkov, W. Cohen, and A. Y. Ng., "Contextual search and name disambiguation in email using graphs," in *Proceedings of SIGIR*, 2006.

[18] B. On and D. Lee, "Scalable name disambiguation using multi-level graph partition," in *Proceedings of SIAM Data Minings*, 2006.

[19] A. Pothen, H. Simon, and K. Liou, "Partitioning sparse sparse matrices with eigenvectors of graphs," *SIAM Journal on Matrix Analysis and Applications*, vol. 11(3), pp. 430–452, 1990.

[20] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, pp. 846–850, 1971.

[21] D. Zeimpekis and E. Gallopoulos, *TMG: A MATLAB toolbox for generating term document matrices from text collections*, 2006.