

# Multilevel $k$ -way Document Clustering : Experiments & Analysis\*

Krishna Gade and George Karypis

University of Minnesota, Department of Computer Science  
Army HPC Research Center, Digital Technology Center  
Minneapolis, MN 55455

{gade, karypis}@cs.umn.edu

## Abstract

In recent years, we have witnessed a tremendous growth in the volume of text documents available on the Internet, digital libraries, news sources and company-wide intranets. This has led to an increased interest in developing methods that can help users to effectively navigate, summarize and organize this information with the ultimate goal of helping them find what they are looking for. Fast and high-quality document clustering algorithms play an important role towards this goal as they have been shown to provide both intuitive navigation/browsing mechanism by organizing large amounts of information into a small number of meaningful clusters as well as greatly improve the retrieval performance either via cluster-driven dimensionality reduction, term-weighting or query expansion. This ever increasing importance of document clustering and the expanded range of its applications led to the development of a number of new and novel algorithms with different complexity-quality tradeoffs. Among them, a class of clustering algorithms that have relatively low computational requirements are those that treat the clustering problem as an optimization process which seeks to maximize or minimize a particular clustering criterion function defined over the entire clustering solution. A common way of performing this optimization is to use a greedy strategy. Such greedy strategies are commonly used in the context of partitional clustering algorithms (e.g.,  $k$ -means), and for many criterion functions it has been shown that they converge to local minima. An alternate way is to use powerful optimizers to do the task and one such technique is the multi-level optimization, which has been effectively used in graph partitioning. In this paper we propose a multilevel optimization technique in the context of document clustering. We experimentally show that multilevel optimizers produce high quality clusters, take very less time and also are scalable to large datasets. Our experiments are conducted with a number of datasets collected from a variety of domains like politics, entertainment and science.

**Keywords:** Document Clustering, Graph Partitioning, Information Retrieval, Multilevel Optimization.

---

\*This work was supported in part by NSF CCR-9972519, EIA-9986042, ACI-9982274, ACI-0133464, and ACI-0312828; the Digital Technology Center at the University of Minnesota; and by the Army High Performance Computing Research Center (AHPCRC) under the auspices of the Department of the Army, Army Research Laboratory (ARL) under Cooperative Agreement number DAAD19-01-2-0014. The content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Access to research and computing facilities was provided by the Digital Technology Center and the Minnesota Supercomputing Institute.

# 1 Introduction

The topic of clustering has been extensively studied in many disciplines and over a variety of different algorithms have been developed. These algorithms can be categorized along different dimensions based on either on the underlying methodology of the algorithm, leading to *agglomerative* [11, 6, 5] or *partitional* [1, 10, 3, 13] approaches, or on the structure of the final solution, leading to *hierarchical* or *non-hierarchical* solutions.

In recent years, various researchers have recognized that partitional clustering algorithms [9, 12] are well-suited for clustering large document datasets due to their low computational requirements. A key characteristic is that they use a global criterion function whose optimization drives the entire clustering process. For some of these algorithms the criterion function is implicit (e.g., PDDP), whereas for other algorithms (e.g., *k*-means) the criterion function is explicit and can be easily stated. This latter class of algorithms can be thought of as consisting of two key components. First is the criterion function optimized by the clustering solution, and second is the actual algorithm that achieves this optimization. These two components are largely independent of each other.

Multilevel paradigm [7, 4] of optimization has been very widely used in problems like graph-partitioning, hypergraph-partitioning with a great deal of success. In multilevel graph partitioning, a given graph is first reduced level by level to a coarser representation, which is partitioned to optimize a given objective function. The partitioned coarser graph is mapped back level-by-level to obtain the partitions in the original graph, by refining the intermediate partitions.

The focus of this paper is to present a multilevel framework in the context of document clustering and evaluate various coarsening schemes of it w.r.t to the traditional greedy (*k*-means) optimization. The rest of the paper is organized as follows. Section 2 provides information on how documents are represented and how the similarity or distance between documents is computed. It also presents the clustering criterion function, which will be used by the clustering algorithms proposed in this paper. Section 3 describes in detail about the two algorithms 1. Greedy Optimization 2. Multilevel Optimization. Section 4 presents experimental results comparing multilevel clustering algorithm with the greedy clustering algorithm along with their analysis. Finally Section 5 provides the conclusive remarks and future work.

## 2 Preliminaries

### 2.1 Document Representation

The various clustering algorithms that are described in this paper use vector-space model to represent each document. In this model, each document  $d$  is considered to be a vector in the term-space, represented by the *term-frequency* (TF) vector

$$d_{tf} = (tf_1, tf_2, \dots, tf_m)$$

where  $tf_i$  is the frequency of the  $i^{th}$  term in the document. A widely used refinement to this model is to weight each term based on its *inverse document frequency*(IDF) in the document collection. The motivation behind this weighting is that terms appearing frequently in many documents have limited discrimination power, and for this reason they need to be de-emphasized. This is commonly done [8] by multiplying the frequency of each term  $i$  by  $\log(N/df_i)$ , where  $N$  is the total number of documents in the collection, and  $df_i$  is the number of documents that contain the  $i^{th}$  term (i.e., document frequency). This leads to the *tf-idf* representation of the document, i.e.,

$$d_{tfidf} = (tf_1 \log(N/df_1), tf_2 \log(N/df_2), \dots, tf_m \log(N/df_m))$$

To account for documents of different length, the length of each document vector is normalized so that it is of unit length ( $\|d_{tfidf}\| = 1$ ), that is each document is a vector in the unit hypersphere. In the rest of the paper, we will assume that the vector representation for each document has been weighted using *tf-idf* and it has been normalized so that it is of unit length.

## 2.2 Similarity Measure

In this paper, we use the popular cosine measure to compute the similarity between a pair of documents. For any two documents  $d_i, d_j$  the cosine similarity is defined as

$$\cos(d_i, d_j) = \frac{d_i^t d_j}{\|d_i\| \|d_j\|}$$

and since the document vectors are of unit length, the above formula simplifies to  $\cos(d_i, d_j) = d_i^t d_j$ . This measure becomes one if the documents are identical, and zero if there is nothing in common between them (i.e., the vectors are orthogonal to each other). From now onwards whenever we say *similarity* of two documents, it means the cosine similarity of the documents.

## 2.3 Definitions

Throughout this paper we will use the symbols  $n, m$  and  $k$  to denote the number of documents, the number of terms, and the number of clusters, respectively. Given a set  $A$  of documents and their corresponding vector representations, we define the *composite* vector  $D_A$  to be

$$D_A = \sum_{d \in A} d$$

and the *centroid* vector  $C_A$  to be

$$C_A = \frac{D_A}{|A|}$$

The composite vector  $D_A$  is nothing more than the sum of all document vectors in  $A$ , and the centroid  $C_A$  is nothing more than the vector obtained by averaging the weights of the various terms present in the documents of  $A$ . Note that even though the document vectors are of length one, the centroid vectors will not necessarily be of unit length.

## 2.4 Vector Properties

By using the cosine function as the measure of similarity between documents we can take advantage of a number of properties involving the composite and centroid vectors of a set of documents. In particular, if  $S_i$  and  $S_j$  are two sets of unit-length documents containing  $n_i$  and  $n_j$  documents respectively, and  $D_i$ , and  $D_j$  and  $C_i$  and  $C_j$  are their corresponding composite and centroid vectors then the following is true:

- The sum of the pair-wise similarities between documents in  $S_i$  and the document in  $S_j$  is equal to  $D_i^t D_j$ . That is,

$$\sum_{d_q \in D_i, d_r \in D_j} \cos(d_q, d_r) = \sum_{d_q \in D_i, d_r \in D_j} d_q^t d_r = D_i^t D_j$$

- The sum of the pair-wise similarities between the documents in  $S_i$  is equal to  $\|D_i\|^2$ . That is,

$$\sum_{d_q, d_r \in D_i} \cos(d_q, d_r) = \sum_{d_q, d_r \in D_i} d_q^t d_r = \|D_i\|^2$$

Note that this equation includes the pairwise similarities involving the same pairs of vectors.

## 2.5 Criterion Function for Document Clustering

At a high-level the problem of document clustering is defined as follows. Given a set  $S$  of  $n$  documents, partition them into a pre-determined number of  $k$  subsets  $S_1, S_2, \dots, S_k$  such that the documents assigned to each subset are more similar to each other than the documents assigned to different subsets. As discussed in the introduction, we could treat clustering as an optimization problem. Consequently, the clustering problem becomes that of given a particular clustering criterion function  $F$ , compute a  $k$ -way clustering solution such that the value of  $F$  is optimized. In this

paper we use a popular criterion function which is used by the  $k$ -means algorithm and which has been shown to give better results in [12]. Maximize Vector-space variant of  $k$ -means algorithm.

$$F = \sum_{r=1}^k \sum_{d_i \in S_r} \cos(d_i, C_r)$$

By making use of the vector properties we can rewrite the above equation as

$$F = \sum_{r=1}^k \| D_r \|$$

Hence, maximizing  $F$  is the same as maximizing the sum of the composite vectors  $D_r$  of each cluster.

### 3 Criterion Function Optimization

#### 3.1 Greedy Optimizer

There are many ways to optimize the above criterion function. A common way of performing this optimization is to use a greedy strategy. Such greedy approaches are commonly used in the context of partitional clustering algorithms (i.e.,  $k$ -means), and for many criterion functions it has been shown that they converge to a local minima. Following is the detailed description of the greedy optimizer we built. Our greedy optimizer consists of two phases:

##### 3.1.1 Initial Clustering

In this phase, a clustering solution is computed as follows. If  $k$  is the number of desired cluster,  $k$  documents are randomly selected to form the *seeds* of these clusters. The similarity of each document to each of these  $k$  seeds is computed, and each document is assigned to the cluster corresponding to its most similar seed. The similarity between documents and seeds is determined using the cosine measure of the corresponding document vectors. This approach leads to an initial clustering solution for the criterion function  $F$ .

##### 3.1.2 Greedy Cluster Refinement

The goal of the cluster refinement phase is to take the initial clustering solution and iteratively refine it. The refinement strategy for  $F$  is as follows. Refinement as said above is an iterative process, and during each iteration, the documents are visited in a random order. For each document  $d_i$ , we compute the change in the value of the criterion function obtained by moving  $d_i$  to one of the other  $k - 1$  clusters. If there exist some moves that lead to an improvement in the overall value of the criterion function, then  $d_i$  is moved to the cluster that leads to the highest improvement. If no such cluster exists  $d_i$  remains in the cluster that it already belongs to. The refinement phase ends, as soon as we perform an iteration in which no documents moved between clusters. Because of its greedy nature of optimizing the solution obtained in each iteration, this refinement strategy always converges to a local minima. Since this algorithm is not guaranteed to converge to a global minima, and the local minima solution highly depends on the particular set of seed documents that were selected in the initial clustering, the overall process is repeated a number of times. That is, we compute  $N$  different clustering solutions (i.e. initial clustering followed by cluster refinement), and the one that achieves the best value for the particular criterion function is kept.

#### 3.2 Multilevel Optimizer

Multilevel optimizer partitions the document term matrix into  $k$  clusters. The basic structure of the algorithm is very simple. The document-term matrix  $M$  is first coarsened down to a few hundred rows (coarsened matrix), a partition of this much smaller matrix is computed, and then this partition is projected back towards the original matrix (finer matrix). At each step of this matrix uncoarsening, the  $k$ -way partition is further refined. Formally, a multilevel clustering algorithm is described here : Consider a document term matrix  $M_0 = (\delta_0, \tau_0)$ , where  $\delta_0$  is the set of

documents and  $\tau_0$  is the set of terms. A pair  $(d, t)$ , where  $d \in \delta_i$  and  $t \in \tau_i$  represents the value of  $t^{th}$  column in the  $d^{th}$  row of the matrix  $M_i$ , for any  $0 \leq i \leq l$ , where  $l$  is the number of levels of coarsening.

- **Coarsening Phase:** The matrix  $M_0$  is transformed into a sequence of smaller matrices  $M_1, M_2, \dots, M_l$  such that  $n_0 > n_1 > n_2 > \dots > n_l$  where  $n_0, n_1, n_2, \dots, n_l$  are the number of rows/documents in the matrices  $M_0, M_1, M_2, \dots, M_l$  respectively.
- **Partitioning Phase** A  $k$ -way partition  $P_l$  of the matrix  $M_l$  is computed that partitions  $\delta_l$  into  $k$  sets.
- **Uncoarsening Phase** The partition  $P_l$  of  $M_l$  is projected back to  $M_0$  by going through intermediate partitions  $P_{l-1}, P_{l-2}, \dots, P_1, P_0$ .

### 3.2.1 Coarsening Phase

During the coarsening phase, a sequence of smaller matrices, each with fewer rows (usually half the preceding number), is constructed. Matrix coarsening can be achieved in various ways. In most coarsening schemes, a set of rows (mostly only two rows) of  $M_i$  are added to form a single row (document) vector of the next level coarser matrix  $M_{i+1}$ . We describe 6 different approaches to carry out the task of coarsening. The central goal of each scheme is to find the most similar document for every document as its match partner. The idea is intuitive from the criterion function  $F$  that tries to maximize the term  $\|D_r\|$  which is nothing more than the square root of the pairwise similarities between all the documents in  $S_r$ .

**Random Coarsening:** In this algorithm the documents are visited in random order and if two documents  $d_i, d_j$  have not been coarsened yet, then we add the two document vectors to form a new document vector  $d_{ij}$  which forms a part of the new coarser matrix. The running time of this algorithm is  $\theta(n)$  for one level of coarsening. Since the number of levels is very negligible when compared to  $n$  for sufficiently large matrices. The total running time of this algorithm is  $O(n)$ .

$$d_{ij} = d_i + d_j$$

**High Similarity Coarsening:** Random coarsening is simple and sometimes efficient depending upon the random choices. But its not guaranteed to coarsen highly similar rows and may sometimes end up coarsening dissimilar rows which might result in a bad clustering. A good coarsening scheme ensures that there are minimum number of moves of documents from one cluster to another during the Uncoarsening and Refinement phase. Thus a good coarsening scheme can potentially make the last phase faster. The high similarity based coarsening algorithm works as follows. The documents are visited in random order and for each document  $d_i$  that is visited, its similarity with the remaining set of the documents in the matrix is calculated and the document  $d_j$  that has the maximum similarity and which has not been visited yet is added to  $d_i$  to form the new document vector  $d_{ij}$  in the next level coarser matrix. As seen above, the algorithm runs in quadratic time to the number of documents present in the matrix at that level. Hence the total runtime is  $\theta(n^2)$ . But this runtime can be improved by employing a simple trick. The trick is for each document  $d_i$  visit only those documents which share atleast one of the terms of  $d_i$ . Since the matrix is sparse in nature as not all documents contain the whole dictionary of words, the algorithm now only takes  $\theta(nnz \times c)$  for one level of coarsening, where  $nnz$  is the total number of non-zero elements in the matrix and  $c$  is a constant which captures the average size of the neighborhood of a document in the matrix, meaning

$$c = \frac{1}{n} \sum_{i=1}^n |N_{d_i}|$$

where  $N_{d_i}$  is the number of documents that have at least one term of  $d_i$ , and  $n$  is the total number of documents in the matrix. So, the total runtime of the algorithm to produce a  $l$ -level coarsening of a matrix is  $O(nnz \times c \times l)$ . which is close to  $O(n)$  for sparse matrices.

**Coarsening by Hashing:** The problem of coarsening a document-term matrix can be modeled as the problem of finding a nearest neighbor for each document and coarsening it with its neighbor. Since finding the exact nearest neighbor is computationally expensive in high dimensional spaces, we settle for an approximate nearest neighbor. Recently some work [2] has been done to solve this problem in the context of association rule mining using a randomized hash based algorithm. We adopt the algorithm to our problem of finding an approximate nearest neighbor for each document vector in the matrix. It is described as follows: Given a document-term matrix, where documents are the rows and terms are the columns, the algorithm first assigns a random value for each column. The random values assigned to the columns are called *hash values* of the column. The matrix formed by the random values is called a *hash matrix*. This completes the preprocessing step of the algorithm. Then, for each document  $d$  in the matrix, we compute its *minhash value*, which is the minimum of hash values assigned to  $d$ 's columns. If two documents (rows) have the same minhash value, then they share a term (column) between each other. After we obtain the minhash value for all the documents, we sort them in the ascending order. Now, for each document  $d$ , we search the sorted list of minhash values for the documents which have the same minhash values. The set of documents that have the same minhash value of  $d$  form the neighborhood of  $d$ . In our implementation, we assigned a constant number  $p$  of random values for each column, so that we have  $p$  different hash matrices instead of one. Hence, we get  $p$  different neighborhoods for each document  $d$ , out of which we select the document which has appeared most number of times (breaking ties arbitrarily) as the approximate nearest neighbor of  $d$ . The approach works well and improves when the  $p$  is increased. The complexity of this approach is divided into three parts : (i) complexity of the preprocessing stage. (ii) complexity of scanning the document-term matrix and sorting the minhashvalues. (iii) searching for the approximate neighbor in the neighborhood of the document. The overall complexity is  $O(m \log m + pn \log n + n\alpha)$  where  $m$  is the number of columns,  $n$  is the number of rows and  $p$  is the number of permutations and  $\alpha$  is the average neighborhood size estimate of each document. The factor of  $\alpha$  which grows in the same rate as  $n$  tends to give quadratic scalabilities w.r.t to the size of the dataset to the algorithm, which we demonstrate later in our experiments.

**Coarsening by Partitioning:** The earlier approaches we have seen (except for the random coarsening) can be unscalable at times, because their computational complexities have a quadratic growth rate w.r.t to the number of documents in the dataset. This approach is specifically designed keeping the scalability factor in mind and the idea behind it is as follows: We randomly partition the documents into equal sized partitions of size  $p$  where  $p \ll n$ . Then for each partition we compute all pair-wise similarities of the documents that belong to a partition. Since the partition size is fixed to be  $p$  we have at most  $\binom{p}{2}$  pair-wise similarities. Now we choose the best match for a document  $d$  within the partition by selecting the one which is most similar with  $d$ . The complexity of this approach is  $\theta(n/p \times p^2) = \theta(np)$ . Since  $p$  is a constant which is very small compared to  $n$ , this approach is linear in the number of documents. Our experiments show that the current approach produces very high quality clustering solutions and is scalable with the size of the dataset.

**Coarsening by Hyperedge Partitioning:** In the above approach we used a scheme of random partitioning. In this approach we attempt to improve the scheme by partitioning the documents in the following manner: A hypergraph  $H = \{V, E\}$  consists of a set of vertices  $\{V\}$  and a set of hyperedges  $\{E\}$ . The concept of a hypergraph is an extension of the concept of a graph in the sense that each hyperedge can connect more than two vertices. In our hypergraph model, we treat the given set of document vectors as vertices in the hypergraph and the set of terms as the hyperedges. Thus, a term (or a column in the document-term matrix)  $t$  is represented by a hyperedge connecting the set of vertices represented by the corresponding set of document vectors  $d$  (or rows in the document-term matrix  $M$ ), such that  $M(d, t) = 1$ . Given this definition, we randomly visit each hyperedge (column) and assign its unvisited vertices (rows) to a single partition. Note that here the partition size ceases to remain constant, so we split those partitions, whose size is greater than a fixed preset threshold into minimal number of partitions each of size at least the fixed threshold. This kind of splitting guarantees the same time bounds shown in the earlier scheme and also guarantees that documents belonging to a partition share at least a column among them.

### 3.2.2 Partitioning Phase

The partitioning phase finds a  $k$ -way clustering or the  $k$ -way partition  $P_l$  of the coarser matrix  $M_l$ . The algorithm is exactly same as the one used in the *greedy optimizer* and which runs in the almost linear time of the number of documents in the coarser matrix.

### 3.2.3 Uncoarsening and Refinement Phase

During the uncoarsening phase, the partition  $P_l$  of the coarser matrix  $M_l$  is projected back to the original matrix, by going through the matrices  $M_{l-1}, M_{l-2}, \dots, M_1$ . Since each document of  $M_{i+1}$  is the sum of a distinct subset of document vectors of  $M_i$ , obtaining  $P_i$  from  $P_{i+1}$  is done by simply assigning the set of documents  $S_i^d$  collapsed/added to  $d \in M_{i+1}$  to the partition  $P_{i+1}[d]$ , meaning

$$P_i[d'] = P_{i+1}[d], \forall d' \in S_i^d$$

. Even though  $P_{i+1}$  is a local minimum clustering of  $M_{i+1}$ , the projected partition  $P_i$  may not be at a local minimum with respect to  $M_i$ . Hence it may still be possible to improve the projected partition. This is achieved by the *greedy refinement* used in the *greedy optimizer*.

## 4 Experiments & Analysis

We experimentally evaluate the above-described 5 coarsening schemes of Multilevel Optimization along with the Greedy Optimization. For convenience sake, we follow a symbolic notation to represent each of the six schemes.

- Greedy Optimization - *G*
- Multilevel Optimization with Random Coarsening - *MR*
- Multilevel Optimization with High Similarity Coarsening - *MS*
- Multilevel Optimization with Hash based Coarsening - *MH*
- Multilevel Optimization with Partiton based Coarsening - *MP*
- Multilevel Optimization with Hyperedge Partition based Coarsening - *MEP*

Every algorithm is run for 10 different random seeds. One seed run of the algorithm conducts 10 trials of optimization and outputs the best(maximum) and average  $F$  values obtained out of the 10 trials along with the total runtime and average time for each trial. For each algorithm we now compute the maximum of maximums, average of maximums and average of averages for the ten seed runs. We also compute the average of the total runtimes for the 10 seed runs and average of average run times for each trial of each seed run. Then, we compare the average of maximums of the multilevel algorithms with the maximum of maximums of the greedy algorithm in Table 2 and also average of averages of the multilevel algorithms with the average of maximums of the greedy algorithm in Table 3. We also compare the average of average run times of the multilevel algorithm with the average of the total runtime of the greedy algorithm. With this kind of comparisons, we show that a single trial of multilevel optimization produces better quality clustering than the best of the 10 trials of greedy optimization and it is faster than the total time needed for the 10 trials of greedy optimization. The following sections describe the datasets used and analysis of the experimental results.

### 4.1 Document Collection

In our experiments, we used a total of 12 different datasets, whose general characteristics are summarized in Table 1. The smallest of these datasets contained 1504 documents and the largest contained 11,162 documents. To ensure diversity in the datasets, we obtained them from different sources. For all data sets, we used a stop-list to remove common words, and the words were stemmed using Porters suffix-stripping algorithm. Moreover, any term that occurs in fewer than two documents was eliminated.

<i>Data</i>	<i>Source</i>	<i>No. of documents</i>	<i>No. of terms</i>	<i>No. of classes</i>
classic	CACM/CISI/CRANFIELD/MEDLINE	7089	12009	4
hitech	San Jose Mercury (TREC)	2301	13170	6
k1a	WebACE	2340	13879	20
la12	LA Times (TREC)	6279	21604	6
mm	Movies/Music	2521	126373	2
new3	TREC	9558	36306	44
ohscal	OHSUMED-233445	11162	11465	10
re0	Reuters-21578	1504	2886	13
re1	Reuters-21578	1657	3758	25
reviews	San Jose Mercury (TREC)	4069	23220	5
sports	San Jose Mercury (TREC)	8580	18324	7
wap	WebACE	1560	8460	20

**Table 1:** Summary of document datasets used in our experiments.

## 4.2 Experimental Methodology and Metrics

For each one of the 12 different datasets we obtained a 10-,20- and 30-way clustering solution optimizing the  $F$  criterion function. We measure the quality of a clustering solution  $F$  measure. Since high  $F$  measures were shown to be capable of producing very low entropy clustering solutions in [12], we concentrate on only  $F$  value as a measure of the quality of the clustering solution i.e., higher the  $F$  value better is the quality of the solution.

## 4.3 Evaluation of $F$ Values produced by Greedy and Multilevel Clustering Schemes

In this section, we evaluate the clustering algorithms with respect to the  $F$  values they produce for the give datasets. For that, we perform two types of comparisons for the  $F$  values:

- Average of the best of 10 trials of Multilevel Optimization for 10 seed runs vs Best of 10 trials of Greedy Optimization for 10 seed runs shown in Table 2
- Average of the average of 10 trials of Multilevel Optimization for 10 seed runs vs Average of 10 trials of Greedy Optimization for 10 seed runs shown in Table 3.

We evaluate the  $F$  values generated by each scheme relative to one another. One way of summarizing the results is to average the  $F$  values for each scheme over the twelve datasets. However, since the clustering quality for different datasets is quite different and since the quality tends to improve as we increase the number of clusters, such a simple averaging may distort the overall results. For this reason, our summarization is based on averaging relative  $F$  values as follows : For each dataset and value of  $k$  (number of clusters), we divided the  $F$  value obtained by a particular clustering scheme by the largest  $F$  value obtained for that particular dataset and  $k$  value over different clustering schemes. These ratios represent the degree to which a particular clustering scheme performed worse than the best clustering scheme for that particular series of experiments. These ratios are less sensitive to the actual  $F$  values and the particular value of  $k$ . We will refer to these ratios as *relative  $F$  values*. Now, for each clustering scheme and value of  $k$  we averaged these relative  $F$  values over the various datasets. A clustering scheme that has an *average relative  $F$  value* close to 1.0 will indicate that this scheme did the best for most of the datasets. On the other hand, if the average relative  $F$  is low, then this scheme performed poorly. Table 4 contain average relative  $F$  values obtained from Table 2 and Table 3 . We can see that the clustering schemes MS,MP perform better, followed by G and MEP. The second tabular form in Table 4 shows that on average, the multilevel clustering schemes MS, MP produce higher  $F$  values for a single trial than the best of the 10 trials produced by the greedy clustering scheme G.

## 4.4 Evaluation of running times of Greedy and Multilevel Clustering Schemes

In this section, we perform a runtime evaluation of the multilevel schemes whose times are averaged for a single run vs 10 runs of Greedy scheme. We are interested in this kind of analysis because earlier in the above section we showed that the Multilevel schemes clearly dominate the greedy scheme in producing high quality clustering solutions for a single run of them compared with 10 runs of greedy. We have done a similar kind of relative analysis as shown

Dataset	Best of Best(Greedy) vs Average of Best(Multilevel): $F$ Values for 10-way Clustering					
	$G$	$MR$	$MS$	$MH$	$MP$	$MEP$
classic	1494.15	1466.80	1490.02	1037.65	1494.05	1484.32
hitech	531.73	530.09	532.71	531.66	532.14	529.04
k1a	552.78	550.30	553.03	551.11	552.39	548.94
la12	1254.42	1242.65	1253.95	1249.70	1255.92	1249.17
mm	614.37	612.32	615.04	614.66	614.72	614.42
new3	2371.96	2356.82	2383.05	2375.47	2383.65	2368.37
ohscal	2143.52	2137.98	2143.34	2143.58	2144.34	2133.29
re0	573.76	572.72	574.06	573.97	573.63	569.05
re1	487.29	487.29	488.86	488.29	489.27	487.83
reviews	1002.09	986.35	1003.91	998.64	1003.44	996.09
sports	2306.72	2297.44	2304.73	2281.91	2306.69	2294.23
wap	395.07	393.93	396.29	395.69	395.88	394.37
Dataset	Best of Best(Greedy) vs Average of Best(Multilevel): $F$ Values for 20-way Clustering					
	$G$	$MR$	$MS$	$MH$	$MP$	$MEP$
classic	1777.33	1758.68	1776.03	-	1782.50	1762.14
hitech	627.73	625.59	630.17	628.36	629.85	626.47
k1a	653.76	650.97	654.99	653.50	654.70	654.01
la12	1464.65	1454.13	1468.43	1465.20	1470.64	1451.61
mm	705.21	701.42	707.76	706.96	706.78	705.63
new3	2844.75	2814.93	2851.09	2838.91	2852.15	2831.93
ohscal	2461.37	2449.73	2463.23	2460.84	2466.61	2460.66
re0	663.42	654.49	665.95	664.86	665.53	660.07
re1	595.74	594.10	598.71	596.33	598.04	593.59
reviews	1125.00	1124.71	1128.24	1125.92	1127.87	1124.38
sports	2677.06	2633.34	2679.95	2658.66	2680.11	2647.11
wap	471.08	470.17	473.19	472.08	472.59	471.63
Dataset	Best of Best(Greedy) vs Average of Best(Multilevel) : $F$ Values for 30-way Clustering					
	$G$	$MR$	$MS$	$MH$	$MP$	$MEP$
classic	1979.05	1954.60	1975.30	1975.88	1980.25	1953.44
hitech	696.63	692.13	699.12	696.48	697.65	694.37
k1a	718.67	716.07	721.32	719.82	720.32	718.95
la12	1617.48	1605.21	1622.88	1619.57	1624.70	1613.91
mm	764.57	759.08	769.18	766.91	768.04	764.21
new3	3143.28	3111.90	3152.43	3134.91	3152.28	3132.03
ohscal	2687.59	2673.43	2693.58	2687.92	2695.12	2685.09
re0	726.27	719.76	728.21	725.78	727.08	723.50
re1	667.22	664.87	670.79	667.48	669.91	667.13
reviews	1212.34	1210.38	1217.36	1216.86	1217.73	1214.98
sports	2887.40	2841.41	2886.58	2866.86	2883.48	2854.77
wap	522.32	518.07	523.00	522.43	521.47	519.50

**Table 2:** Comparison of Best  $F$  of 10 runs of Best  $F$  of 10 seeds for Greedy Scheme ( $G$ ) with Average  $F$  of 10 runs of Best of 10 seeds for 5 different Multilevel Schemes.

in Section 4.3. For each dataset and value of  $k$  (number of clusters), we divided the running time for a particular clustering scheme by the lowest running time for that particular dataset and  $k$  value over different clustering schemes. These ratios represent the degree to which a particular clustering scheme performed worse than the best clustering scheme for that particular series of experiments. These ratios are less sensitive to the actual running times and the particular value of  $k$ . We will refer to these ratios as *relative running times*. Now, for each clustering scheme and value of  $k$  we averaged these relative running times over the various datasets. A clustering scheme that has an *average relative running time* close to 1.0 will indicate that this scheme was the fastest for most of the datasets. On the other hand, if the average relative running time is higher, then this scheme was slower. Table 6 contain average relative  $F$  values obtained from Table 5. We can see that the clustering schemes  $MS$ ,  $MP$  and  $MEP$  were the faster, while  $G$  was the slowest. The Table 6 shows that a single trial of all multilevel schemes is faster than 10 trials of the greedy scheme.

Thus out of the five multilevel schemes, only  $MS$  and  $MP$  dominate  $G$  in both quality and runtime. Before reaching a conclusion about which one of  $MS$  and  $MP$  is better, we do a scalability study for all the multilevel algorithms in the next section. Although both  $MS$  and  $MP$  produce high quality clusterings compared to  $G$  only  $MP$  is linearly scalable where as  $MS$  has quadratic scalability.

Dataset	Average of Best(Greedy) vs Average of Average(Multilevel) : F Values for 10-way Clustering					
	G	MR	MS	MH	MP	MEP
classic	1492.03	1446.42	1479.18	1425.31	1489.74	1484.32
hitech	530.19	527.23	530.88	529.27	530.92	529.04
k1a	549.90	545.49	550.87	547.39	550.36	548.94
la12	1252.99	1234.67	1250.38	1242.43	1252.42	1249.17
mm	611.05	604.40	613.86	611.25	612.37	614.42
new3	2366.04	2332.66	2373.95	2356.32	2371.47	2368.37
ohscal	2140.99	2129.87	2134.34	2135.28	2141.03	2133.29
re0	572.41	565.34	572.37	572.46	572.37	569.05
re1	484.74	484.44	486.73	484.84	487.24	487.83
reviews	1000.00	972.48	1000.96	980.53	1000.33	996.09
sports	2302.13	2277.09	2288.90	2249.13	2302.47	2294.23
wap	394.41	391.64	394.74	393.88	394.81	394.37
Dataset	Average of Best(Greedy) vs Average of Average(Multilevel) : F Values for 20-way Clustering					
	G	MR	MS	MH	MP	MEP
classic	1770.65	1746.98	1766.80	-	1774.19	1762.14
hitech	626.85	622.56	628.35	625.89	627.97	626.47
k1a	651.57	647.92	653.22	650.30	652.66	654.01
la12	1461.95	1448.00	1463.60	1458.29	1465.98	1451.61
mm	701.68	696.34	705.27	703.52	703.33	705.63
new3	2829.89	2790.08	2841.12	2818.02	2844.07	2831.93
ohscal	2457.66	2441.90	2456.12	2453.43	2461.08	2460.66
re0	660.73	649.80	663.04	661.19	662.90	660.07
re1	593.28	590.96	596.21	592.57	596.08	593.59
reviews	1121.12	1119.27	1124.91	1122.07	1124.71	1124.38
sports	2667.48	2611.48	2668.93	2636.99	2669.74	2647.11
wap	469.66	467.40	471.76	469.93	471.29	471.63
Dataset	Average of Best(Greedy) vs Average of Average(Multilevel) : F Values for 30-way Clustering					
	G	MR	MS	MH	MP	MEP
classic	1962.43	1941.48	1964.01	1962.22	1973.29	1953.44
hitech	694.58	689.43	697.05	694.00	695.14	694.37
k1a	715.58	711.75	718.81	716.75	717.54	718.95
la12	1612.34	1598.43	1617.70	1612.31	1619.08	1613.91
mm	762.20	753.89	767.01	763.59	764.76	764.21
new3	3133.64	3096.69	3142.35	3120.71	3143.09	3132.03
ohscal	2682.75	2666.10	2685.59	2678.14	2689.09	2685.09
re0	724.21	714.72	726.02	722.75	724.75	723.50
re1	665.01	661.19	668.01	663.67	666.17	667.13
reviews	1207.82	1205.12	1213.15	1211.64	1214.08	1214.98
sports	2877.70	2820.57	2874.45	2854.06	2873.68	2854.77
wap	519.69	515.85	521.62	520.25	519.81	519.50

**Table 3:** Comparison of Average  $F$  of 10 runs of Best  $F$  for 10 seeds for Greedy Scheme (G) with Average  $F$  of 10 runs of 10 seeds for 5 different Multilevel Schemes.

## 4.5 Scalability Analysis

This is perhaps the most important analysis before deciding the best scheme of the multilevel partitioning. Scalability analysis is conducted for all coarsening schemes. We have chosen *sports* dataset to study the scalability of the multilevel schemes and we replicate it 5 times, 10 times, 15 times all the way upto 20 times. We run the coarsening schemes on each one of the replicas and then tabulate the run times. The scalability table for the *sports* dataset is give below. The size of the sports dataset is 8580 documents to start with and its increased in size as shown in the Table 7.

As shown from the table the MS and MH schemes have quadratic scalabilities. We predicted this before in our theoretical analysis that the schemes High Similarity Matching and Hash Based Scheme might have quadratic growth rates and our experiments show the same. Hence these two techniques although good in terms of producing better  $F$  values proved to be unscalable to larger datasets.

$k$	Average Relative $F$ Values from Table 2						$k$	Average Relative $F$ Values from Table 3					
	$G$	$MR$	$MS$	$MH$	$MP$	$MEP$		$G$	$MR$	$MS$	$MH$	$MP$	$MEP$
10	0.998	0.992	<b>0.999</b>	0.972	<b>1.000</b>	0.994	10	<b>0.998</b>	0.986	<b>0.998</b>	0.967	<b>0.999</b>	0.997
20	0.997	0.990	<b>0.999</b>	0.997	<b>1.000</b>	0.993	20	<b>0.997</b>	0.987	<b>0.999</b>	0.995	<b>0.999</b>	0.996
30	0.997	0.989	<b>1.000</b>	0.988	<b>0.999</b>	0.993	30	0.996	0.987	<b>0.999</b>	0.987	<b>0.999</b>	0.996

**Table 4:** Relative  $F$  values from Table 2 and Table 3 averaged over the different datasets for the greedy and 5 different multilevel schemes. Boldfaced entries represent the best and the second best performing schemes.

Dataset	Average Run Times for 10-way Clustering						Dataset	Average Run Times for 10-way Clustering					
	$G$	$MR$	$MS$	$MH$	$MP$	$MEP$		$G$	$MR$	$MS$	$MH$	$MP$	$MEP$
classic	6.48	9.46	6.29	6.38	7.20	6.21	classic	17.38	20.04	12.88	0.00	15.68	13.39
hitech	9.13	11.96	8.51	19.50	9.47	9.45	hitech	32.75	26.64	18.31	32.90	21.60	20.61
k1a	14.09	14.16	8.55	12.25	9.86	9.32	k1a	37.42	30.34	18.24	24.94	22.27	20.38
la12	55.62	45.70	31.38	40.08	35.25	33.81	la12	141.73	92.25	65.08	82.14	76.60	70.03
mm	17.83	16.91	12.03	23.96	14.11	13.31	mm	58.45	40.34	29.55	45.56	33.48	30.73
new3	86.34	69.85	54.35	63.16	54.31	49.52	new3	271.38	149.24	104.28	134.07	120.33	107.48
ohscal	31.29	23.24	19.43	23.24	19.51	19.83	ohscal	90.15	56.94	45.59	54.86	51.14	49.69
re0	1.04	0.73	0.55	0.68	0.57	0.52	re0	2.38	1.53	1.05	1.23	1.19	1.08
re1	1.32	1.00	0.67	0.95	0.77	0.73	re1	4.39	3.59	2.21	2.77	2.86	2.44
reviews	27.89	27.67	18.21	31.17	19.18	18.36	reviews	97.87	59.45	38.57	61.99	48.75	43.46
sports	34.55	33.48	23.55	38.12	24.51	23.60	sports	107.42	77.89	51.28	74.95	62.56	56.31
wap	5.67	5.74	3.42	4.70	3.98	3.78	wap	22.38	16.50	10.58	13.49	13.01	11.92

Dataset	Average Run Times for 10-way Clustering					
	$G$	$MR$	$MS$	$MH$	$MP$	$MEP$
classic	32.36	25.68	18.59	23.69	23.01	20.66
hitech	63.48	34.57	23.90	41.49	29.65	27.60
k1a	63.96	37.73	24.88	31.91	30.36	27.10
la12	224.32	134.56	90.82	117.75	109.38	104.24
mm	103.53	54.48	41.27	60.17	47.34	45.25
new3	469.24	164.69	142.58	189.09	179.31	159.42
ohscal	157.25	87.01	68.95	81.82	80.60	76.32
re0	6.64	4.69	2.73	3.30	3.44	2.89
re1	12.04	9.22	5.61	6.94	6.85	6.41
reviews	172.15	80.85	58.15	84.46	73.25	67.99
sports	204.11	113.95	79.77	107.64	100.13	90.71
wap	39.50	24.21	16.36	19.57	19.01	17.99

**Table 5:** Comparison of the Average Running Times over 10 runs of all the schemes.

## 4.6 Discussion

Before continuing further discussion, we present the 20-way, 30-way clustering results of comparison between greedy clustering algorithm ( $G$ ) and multilevel clustering with partition-based coarsening algorithm ( $MP$ ) in Table 8. For these experiments, we tuned the number of trials of optimization taken by both  $G$  and  $MP$  such that their runtimes are roughly equal. One can observe from the above results that the multilevel scheme outperforms (although by a small margin) the greedy scheme for most of the datasets. This test, further corroborates that given the same time frame, multilevel solutions are superior to their greedy counterparts in producing higher  $F$  values.

### 4.6.1 Greedy Optimization

Greedy scheme optimizes the criterion function pretty much on par with that of multilevel schemes, but fails to do that in comparable time. Greedy fails to producing high quality clustering solutions because of two reasons. (i) Initial seed

$k$	Average Relative Run Times from Table 5					
	$G$	$MR$	$MS$	$MH$	$MP$	$MEP$
10	1.583	1.464	<b>1.014</b>	1.494	<b>1.110</b>	<b>1.051</b>
20	2.077	1.487	<b>1.000</b>	1.352	<b>1.195</b>	<b>1.084</b>
30	2.502	1.435	<b>1.000</b>	1.334	<b>1.220</b>	<b>1.119</b>

**Table 6:** Relative running times from Table 5 averaged over the different datasets for the greedy and 5 different multilevel schemes. Boldfaced entries represent the fastest, second fastest and third fastest performing schemes.

Size of the Sports Dataset	Running Time				
	MR	MS	MH	MP	MEP
1x	2.19	<b>8.3</b>	<b>13.22</b>	3.46	4.73
5x	16.09	<b>103.3</b>	<b>112.88</b>	27.44	15.6
10x	31.72	<b>529.63</b>	<b>434.1</b>	58.82	42.07
15x	71.98	<b>1612.26</b>	<b>1322.67</b>	125.33	125.33
20x	106.51	<b>3399.76</b>	<b>3243.08</b>	152.21	178.35

**Table 7:** Scalability of the multilevel schemes w.r.t to the matrix size

Dataset	Best $F$ : 20-way		Running Times		Dataset	Best $F$ : 30-way		Running Times	
	G	MP	G	MP		G	MP	G	MP
classic	1766.670	1783.790	14.580	14.660	classic	1969.960	1978.000	23.770	21.460
hitech	626.010	628.660	20.520	20.520	hitech	695.140	694.840	29.810	31.330
k1a	653.240	653.480	19.490	20.380	k1a	713.330	721.420	29.940	27.280
la12	1463.620	1468.000	69.060	68.820	la12	1612.710	1612.380	112.090	98.650
mm	703.640	702.200	26.380	28.810	mm	759.280	767.800	46.150	44.470
new3	2826.260	2850.630	134.230	112.760	new3	3129.070	3143.650	177.740	178.730
ohscal	2451.330	2459.810	48.200	44.140	ohscal	2678.310	2695.700	68.310	69.110
re0	660.770	661.600	1.000	1.140	re0	723.830	724.310	3.160	3.200
re1	591.760	596.930	2.430	2.330	re1	662.340	667.880	5.330	6.530
reviews	1117.460	1122.650	64.560	46.970	reviews	1205.980	1212.420	76.240	66.410
sports	2669.510	2660.560	57.450	61.040	sports	2880.850	2866.420	97.100	93.470
wap	468.260	469.820	9.800	11.490	wap	518.900	521.050	33.570	35.230

**Table 8:** Comparison of Greedy (G) and Multilevel (MP) schemes for  $F$  values given that both G and MP take roughly the same amount of running time.

selection plays a major role in the success of an EM-based algorithm. And since the initial seed selection is done in a random manner, greedy fails to produce high quality clusters. (ii) Cluster refinement approach is greedy in nature, as it moves documents from a cluster to another if there is an immediate increase in the objective function.

#### 4.6.2 Multilevel Optimization

Multilevel to some extent solves the first problem of Greedy approach because of coarsening the matrix. Since a good coarsening will create a better coarser matrix which makes it easier to reduce the error in the random seed selection there. Also, successive refinements tend to improve the solution level by level. Lets discuss in detail regarding each coarsening scheme. *Random Coarsening* fails to produce high quality clusters because of bad coarser matrix which has been formed by merging dissimilar documents. This is the reason why it also takes longer times, because bad coarsening makes successive refinements slower. *Similarity Based Coarsening* produces high quality clusters and its running time is also very good, but it grows quadratically with the problem size. It is because of the fact that this algorithm searches the entire neighborhood of the document, for each document. So, if we double the problem size both the documents and their neighborhoods tend to grow together and hence we have a quadratic growth rate. *Hash Based Coarsening* does a moderate job in producing quality clusters and also is a bit on the slower side because of its search operation in the sorted minhash value list for each document. It is also not linearly scalable for the same reason specified for similarity based coarsening. *Partition Based Coarsening* emerges as the final winner performing very well in all the three characteristics. Its runtime is linear in the number of documents and also it scales well for larger datasets. *Hyperedge Partition Based Coarsening* surprisingly falls behind the naive partition based coarsening. Since, the term space is huge for document datasets, choosing only one hyperedge (column) in determining the groups of documents that go into the same partition, may not be working very well compared to a random partition scheme. It might improve if a constant number ( $> 1$ ) of hyperedges are chosen while assigning the partitions.

## 5 Conclusion and Future Work

In this paper we introduced the multilevel paradigm of optimizing criterion functions for clustering documents. We came up with 5 different coarsening schemes to implement in this algorithm. We evaluated the performance of these 5 techniques both w.r.t one another and also w.r.t the greedy optimization ( $k$ -means type). We have experimentally

shown that multilevel schemes outdo the greedy scheme in optimizing the criterion function (though not by a great margin) and some of them are faster than the greedy scheme by a factor of 2. We have introduced a new class of coarsening techniques called *Partition based Coarsening* schemes and eventually showed that these schemes tend to produce high quality clusters and consume very less time and are highly scalable. In future we intend to apply the multilevel framework to other clustering criterion functions and improve the existing coarsening techniques. A straightforward extension can be to apply this framework to the *recursive bisection* class of clustering algorithms.

## References

- [1] D. Boley, M. Gini, R. Gross, E.H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems (accepted for publication)*, 1999.
- [2] Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, and Jeffrey Ullman. Finding interesting associations without support pruning. In *IEEE Transactions on Data Engineering*, 2001.
- [3] Inderjit S. Dhillon, Yuqiang Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. pages 131–138, Maebashi City, Japan, 2002.
- [4] Vipin Kumar George Karypis, Rajat Aggarwal and Shashi Shekhar. Multilevel hypergraph partitioning: Applications in vlsi domain. In *34th Design Automation Conference*, 1998.
- [5] G. Karypis, E.H. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.
- [6] G. Karypis, E.H. Han, and V. Kumar. Multilevel refinement for hierarchical clustering. Technical Report TR-99-020, Department of Computer Science, University of Minnesota, Minneapolis, 1999.
- [7] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. In *SIAM Journal on Scientific Computing*, 1995.
- [8] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.
- [9] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.
- [10] Y. Zhao and G. Karypis. Comparison of agglomerative and partitional document clustering algorithms. In *SIAM(2002) workshop on Clustering High-dimensional Data and Its Applications*, 2002. Also available as technical report #02-014, university of Minnesota.
- [11] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proc. of Int'l. Conf. on Information and Knowledge Management*, pages 515–524, 2002.
- [12] Ying Zhao and George Karypis. Criterion functions for document clustering: Experiments and analysis. Technical Report TR #01–40, Department of Computer Science, University of Minnesota, Minneapolis, MN, 2001. Available on the WWW at <http://cs.umn.edu/~karypis/publications>, to be published in the Journal of Machine Learning.
- [13] Shi Zhong and Joydeep Ghosh. A comparative study of generative models for document clustering. In *SIAM Int. Conf. Data Mining Workshop on Clustering High Dimensional Data and Its Applications*, San Francisco, CA, May 2003.