Clique percolation for finding naturally cohesive and overlapping document clusters

Wei GAO
Singapore Management University, weigao@smu.edu.sg

Kam-Fai WONG

Yunqing XIA

Ruifeng XU

Clique Percolation Method for Finding Naturally Cohesive and Overlapping Document Clusters

Wei Gao, Kam-Fai Wong, Yunqing Xia, and Ruifeng Xu

Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong, China
{wgao, kfwong, yqxia, rfxu}@se.cuhk.edu.hk

Abstract. Techniques for find document clusters mostly depend on models that impose strong explicit and/or implicit priori assumptions. As a consequence, the clustering effects tend to be unnatural and stray away from the intrinsic grouping natures of a document collection. We apply a novel graph-theoretic technique called *Clique Percolation Method* (CPM) for document clustering. In this method, a process of enumerating highly cohesive maximal document cliques is performed in a random graph, where those strongly adjacent cliques are mingled to form naturally overlapping clusters. Our clustering results can unveil the inherent structural connections of the underlying data. Experiments show that CPM can outperform some typical algorithms on benchmark data sets, and shed light on its advantages on natural document clustering.

1 Introduction

Clustering is an important technique that facilitates the navigation, search and analysis of information in large unstructured document collections. It is an unsupervised process to identify inherent groupings of similar documents, where documents exhibit high intra-cluster similarity and low inter-cluster similarity.

Many existing clustering algorithms optimize criterion functions with respect to the employed similarity measures over all the documents assigned to each possible partition of the collection [10,19]. They always impose some explicit and/or implicit constraints as to the number, size, shape or disjoint characteristics of target clusters. For example, partitional algorithms like k-means assume cluster number k and do not allow one document belonging to multiple groups. Although fuzzy clustering, such as fuzzy C-means algorithm [2,12], does support overlapping clusters by a membership function and a fuzzifier parameter, they are still confined by cluster number and can find only spherical shape clusters. Some algorithms are model-based, e.g., Naive Bayes or Gaussian Mixture model [1,13]. They assume certain probabilistic distributions of the documents and try to find a model maximizing the likelihood of data. When data cannot fit the presumed distribution, poor cluster quality can result. k-way clustering or bisection algorithms [19] force clusters to be equally sized. Spectral clustering [7,8] has emerged as one of the most effective clustering tools based on max-flow/min-cut theorem [4]. However, they prohibit overlapping clusters.

We define natural document clustering as a problem of finding unknown number of overlapping as well as cohesive document groups with varied sizes and arbitrary distributions of the data. We try to obtain the clustering results with these free characteristics by reducing as many external constraints as feasible and leaving things to the inherent grouping nature among documents. For this purpose, we propose a document clustering technique using a novel graph-theoretic algorithm, named Clique Percolation Method (CPM). The idea is to identify adjacent maximal complete subgraphs, which is referred to as Maximal Document Cliques (MDC), in the document similarity graph using a threshold clique, and then mingle those strongly adjacent MDCs to form naturally overlapping document clusters. Although it does introduce an explicit parameter k, which is the size of the threshold clique, our algorithm can automatically settle the critical point, at which the natural clustering can be achieved. We show that CPM outperforms representative clustering methods with experiments on the benchmark data.

The rest of this paper is organized as follows: Section 2 describes the proposed CPM; Section 3 presents the algorithmic implementation of this technique; Section 4 gives related work; Section 5 presents experimental evaluation results; Finally, we conclude this paper.

2 Document Clustering by Clique Percolation Method

2.1 Preliminaries

In general, suppose $V = \{d_1, d_2, \ldots, d_{|V|}\}$ is a collection of documents. We represent the collection by an undirect graph $G = (V, E)$, where V is the vertex set and E is the edge set such that each edge $\{i, j\}$ is a set of two adjacent vertices d_i, d_j in V. The adjacent matrix M of the graph is defined by $M = [m_{ij}]_{i,j=1}^{|V|}$, where each entry w_{ij} is the edge weight which is the value of similarity metric (in what follows we use Cosine coefficient) between d_i and d_j. The graph can also be unweighted where an edge exists indicating the distance of its two vertices smaller than some threshold, in which case w_{ij} is binary.

A *clique* in G is a subset $S \subseteq V$ of vertices, such that $\{i, j\} \in E$ for all distinct $\{d_i, d_j\} \in S$. Thus any two vertices are adjacent in a clique that constitutes a complete subgraph of G. A clique is said to be *maximal* if its vertices are not a subset of the vertices of a larger clique, which is referred to as a Maximal Document Clique (MDC) in a document similarity graph. MDC is considered the strictest definition of a cluster [15]. In graph theory, enumerating all maximal cliques (equivalently, all maximal independent sets or all minimal vertex covers) is believed NP-hard [3,18].

Suppose $|V|$ number of documents are given in a measure space with a similarity metric w_{ij}. We define a binary relation $\sim_t$ between documents on $G = \{V, E\}$ with respect to parameter t: $i \sim_t j := w_{ij} \leq t$, which is self-reflexive, symmetric and non-transitive. There is an edge $\{i, j\} \in E$ connecting vertices d_i and d_j whenever $i \sim_t j$ with respect to threshold t. Figure 1 illustrates that given a

	1	2	3	4	5	6	7
1	0	4	5	4	2	2	3
2	4	0	3	4	4	6	4
3	5	3	0	7	5	7	6
4	4	4	7	0	2	4	2
5	2	4	5	2	0	2	1
6	2	6	7	4	2	0	2
7	3	4	6	2	1	2	0

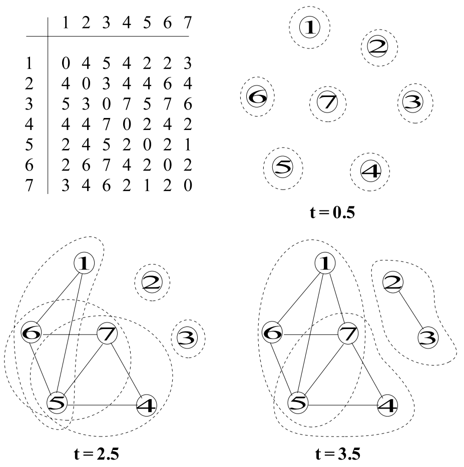

Fig. 1. Graphs with respect to threshold level t and different cohesive MDC clusters (in dotted regions) resulted from it

matrix reflecting the distances between 7 documents and the t value, a series of graphs for the relation $i \sim_t j$ are produced with different connectivity densities. Clearly, if each MDC is considered as a cohesive form of cluster, we can discover different number of clusters from these graphs, where $t = 0.5$, 2.5 and 3.5 results in 7, 5 and 3 number of clusters, respectively. They display interesting properties of natural clusters except for excessive intra-cluster cohesiveness.

The series of graphs parameterized by t above can be seen as random graphs with constant set of vertices and a changing set of edges generated with probability p, the probability two vertices can be connected by an edge. Intuitively, tuning the value of t is somehow equivalent to adding or removing some edges according to p in monotonic manner. In order for an appropriate t, we first determine p_c, the critical value of p, and then derive t from p_c by making use of their interdependency relationship. The critical value p_c is defined as the probability, under which a giant k-clique percolation cluster will emerge in the graph, and is known as the percolation threshold for a random network [6]. At this threshold, the percolation transition takes place (see Section 2.2). For clustering, the assumption behind is that no cluster can be excessively larger than others, which can be achieved by commanding $p < p_c$.

2.2 Clique Percolation Method in Random Graphs

Concepts of *k*-Clique Percolation. The concept of k-clique percolation is fundamental for Clique Percolation Method (CPM) in random networks, which was studied in [6]. The successful applications of CPM for uncovering community structure of co-authorship networks, protein networks and word association graphs can be found in [14]. Hereby we briefly present some related notions.

Definition 1. *k-clique is defined as a complete subgraph of k vertices.*

Definition 2. *k-clique adjacency: Two k-cliques are adjacent if they share $k-1$ vertices, i.e., if they differ only in a single vertex.*

Definition 3. *k-clique percolation cluster is a maximal k-clique-connected subgraph, i.e., it is the union of all k-cliques that are k-clique adjacent. Obviously, a k-clique percolation cluster is unnecessarily a MDC, but it must be equivalent to the union of all MDCs adjacent by at least $k-1$ vertices.*

Definition 4. *k-clique adjacency graph is the compressed form of the original graph, where the vertices denote the k-cliques of the original graph and there is an edge between two vertices if the corresponding k-cliques are adjacent.*

Moving a particle along an edge on a k-clique adjacency graph is equivalent to rolling a k-clique template (threshold clique) from one k-clique on the original graph to an adjacent one. A k-clique template can be placed onto any k-clique of the original graph, and rolled to an adjacent k-clique by relocating one of its vertices and keeping other $k-1$ vertices fixed. Thus, the k-clique percolation clusters are all those subgraphs that can be fully explored by rolling a k-clique template in them [6]. Note that a k-clique percolation cluster consists of all MDCs adjacent by at least $k-1$ vertices. Thus, the cohesiveness of documents in a k-clique percolation cluster as well as the overlap degree between clusters can be tuned by the k value. The goal of CPM is to find all k-clique percolation clusters.

Percolation Threshold p_c. How to estimate the threshold probability p_c of k-clique percolation with respect to k ($k \geq 2$)? The clique percolation theory emphasizes that under such p_c (critical point), a giant k-clique percolation cluster that is excessively larger than other clusters will take place [9,6]. Intuitively, the greater the p ($p > p_c$) is, the more likely the giant cluster appears, and the larger its size (which includes most of graph nodes), as if using a k-clique can percolate the entire graph.

Consider the heuristic condition of template rolling at the percolation threshold: after rolling a k-clique template from a k-clique to an adjacent one by relocating one of its vertices, the expectation of the number of adjacent k-cliques, where the template can roll further by relocating another of its vertices, be equal to 1. The intuition behind is that a larger expectation value would allow an infinite series of bifurcations for the rolling, ensuring that a giant cluster is present in the graph. The expectation value can be estimated as $(k-1)(|V|-k)p_c^{k-1} = 1$, where $(k-1)$ is the number of template vertices that can be selected for the next relocation, $(|V|-k)$ is the number of potential destinations for this relocation, out of which only the fraction p^{k-1} is acceptable, because each of the new $k-1$ edges must exist in order to reach a new k-clique after relocation. Therefore, the percolation threshold function $p_c(k)$ with respect to k and $|V|$ is as follows:

$$p_c(k) = [(k-1)(|V|-k)]^{-\frac{1}{k-1}} \tag{1}$$

Generation of Random Graph. According to Eq. (1), we can obtain a series of critical values with regard to the threshold clique sizes provided, which are actually the threshold probabilities of connecting two document vertices by an edge at these critical points. How to generate a random graph with the exactly desirable connectivity is technically very challenging since the degree distribution of each vertex needs to be appropriately modeled. Some work on systematically modeling degree distribution has been done in the field of random networks [9]. In this study, we prefer to simplify our specific problem by using two heuristics.

First, for each vertex, we consider its N-Nearest Neighbors (NNB) instead of using a fixed similarity threshold value, where N is determined by the formula:

$$N(k) = p_c(k) \times \frac{|V| - 1}{k - 1} \tag{2}$$

where the factor $\frac{|V|-1}{k-1}$ actually scale the size of the original graph down to the level of k-cliques in the graph and $p_c(k)$ is considered as the average proportion of k-cliques are the nearest neighbors of a given clique in the k-clique adjacency graph. Here we actually use the connectivity of the k-clique adjacency graph to simulate the original graph. The reason we don't use $N(k) = p_c(k) \times (|V| - 1)$ is because the generated graph tends to be over dense since $p_c(k)$ is not the proportion of NNB nodes, but in fact the probability of two vertices being connected by an edge.

Secondly, we examine the co-relation between p and the similarity threshold t. Given p_c, we can estimate the bound(s) of t_c so that the graph with the approximated connectivity as that under p_c could be generated. Because p-t are monotone, a graph could be produced with edge weights t greater than t_c. We derive t_c by a simple approximation:

$$t_c(k) = 0.5 + p_c(k) \times (w_{max} - w_{min}) \tag{3}$$

where w_{max} and w_{min} are the maximum and minimum values of document similarity in the collection, respectively. Intuitively, we deem that only edge weights somewhat larger than 0.5 are considered similar. We also observe $p_c(k)$ is well below 0.5 for the normal size of corpus as k is not too large (< 20), which can be shown in Fig. 2 and guarantees $t_c \leq 1$.

We then apply the two heuristics incrementally during the graph generation process, i.e. by generating connections between NNBs for each vertex at first place and then prune the edges with weights less then t_c. The intuition is that denser graphs are penalized more heavily by the combination of the two heuristics.

3 Algorithmic Implementation of CPM

The clustering process is turned out to be a problem of finding all MDCs and then merging those with at least $k-1$ common nodes into clusters. The proposed CPM clustering algorithm includes 5 major steps:

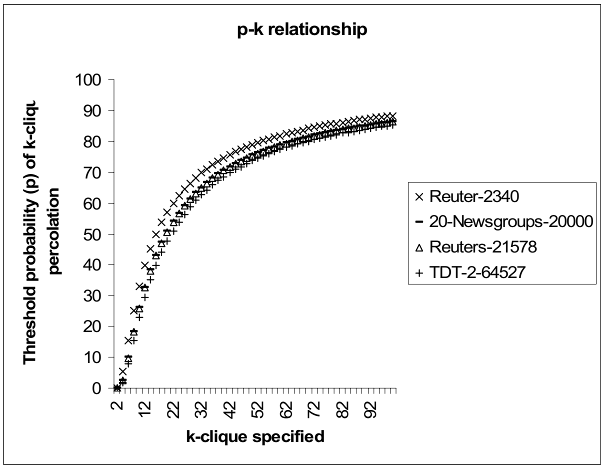

Fig. 2. The curve of p-k relationship given typical sizes of benchmark corpora indicates p_c is below 0.5 when $k < 20$

1. Preprocessing: Eliminate words in the stop list, use Porter's stemmer as the stemming algorithm, build document vectors, and create a $|V| \times |V|$ document similarity matrix $\mathbf{A}$;
2. Given k as parameter, compute Eq. (1) for $p_c(k)$, Eq. (2) for $N(k)$ and Eq. (3) for $t_c(k)$;
3. Create document similarity graph G from $\mathbf{A}$, where each vertex is connected with $N(k)$ NNBs. For each edge, prune it if its weight $w_{ij} < t_c(k)$;
4. Enumerate all MDCs in G using Algorithm 1;
5. Create a $M \times M$ adjacent matrix $\mathbf{B}$ (where M is the number of MDCs), find k-clique percolation clusters using Algorithm 2 on $\mathbf{B}$.

Enumerating Maximal Document Cliques. Algorithms for finding maximal cliques (step 4) were studied in [3] and achieved processing time bounded by $O(v^2)$ where v is the number of maximal cliques. Their algorithms are distinctive because they can be applied to a graph of comparatively large size. We implement an efficient counterpart of the algorithm using back-tracking method (see Algorithm 1). A MDC is output at each end of back-track. The running time is $O(v)$.

Finding *k*-Clique Percolation Clusters. When all the MDCs are enumerated, a clique-clique adjacent matrix is prepared. It is symmetric where each row and column represents a MDC and the entries are the number of common vertices between two cliques (the diagonal values are the sizes of MDCs). The k-clique percolation clusters are one-to-one correspondent to the connected components in the clique-clique adjacency graph represented by the matrix, which can be obtained using Algorithm 2 (step 5 above). The algorithm first

Algorithm 1. Enumerate All MDCs

input: Vertex set V and edge set E of graph G.
output: All MDCs of G into C.

```
procedure EnumMDC ($C$, $U$, $E$)
 1: if ($U = \phi$) then
 2:     output $C$
 3:     return
 4: end if
 5: for every vertex $u \in U$ do
 6:     $U := U - \{u\}$
 7:     EnumMC ($C \cup \{u\}, U \cap \{v|(v,u) \in E\}$)
 8: end for
end procedure

$C := \phi$
EnumMDC ($C$, $V$, $E$)
```

creates a clique-clique adjacent matrix **B**, in which every off-diagonal entry smaller than $k-1$ and every diagonal element smaller than k are erased (line 2–12), and then carrying out a depth-first-search (DFS) to find all the connected components.

4 Related Work

Traditional hierarchical agglomerative clustering (HAC) are intrinsically graph-based like CPM. HAC treats each data point as a singleton cluster and then successively merges pairs of clusters until all clusters have been merged into a single cluster that contains all documents. Single-link, complete-link and average-link are the most popular HAC algorithms.

In single-link algorithm [16], the similarity between clusters is measured by their most similar members (minimum dissimilarity). Generally, agglomerative process are rather computationally intensive because the minimum of inter-cluster distances must be found at each merging step. For single-link clustering, an efficient implementation of Minimum Spanning Tree (MST) algorithms of a weighted graph is often involved. Therefore, single-link produces clusters that are subgraphs of the MST of the data and are also connected components. It is capable of discovering clusters of varying shapes, but often suffers from the so-called chaining effect. Complete-link [11] measures the similarity between two clusters by their least similar members (maximum dissimilarity). From graph-theoretic perspective, complete-link clusters are non-overlapping cliques and are related to the node colorability of graphs. Complete-link is not vulnerable to chaining effect, but generates excessive compact clusters and is thus very sensitive to outliers. Average-link clustering [5] is a compromise between single-link

Algorithm 2. Find All k-Clique Percolation Clusters

input: A set of all MDCs C and k.
output: All k-clique percolation clusters into P.

```
procedure Find-k-CPC ($P$, $C$, $k$)
 1: $B := 0$ //Initialize $B$'s elements as 0
 2: for $i$ from 1 to $M$ do
 3:    for $j$ from 1 to $M$ do
 4:       $B[i][j] := |C_i \cap C_j|$ //# of common nodes of two MDCs
 5:       if $(i = j) \wedge (B[i][j] < k)$ then
 6:          $B[i][j] := 0$ //Off-diagonal element $< k$ is replaced by 0
 7:       end if
 8:       if $(i \neq j) \wedge (B[i][j] < k - 1)$ then
 9:          $B[i][j] := 0$ //Diagonal element $< k - 1$ replaced by 0
10:       end if
11:    end for
12: end for
13: $P := \phi$; $i := 1$ //Initialize output container $P$ and recursion counter $i$
14: DFS($P$,$B$,$i$) //DFS to output connected components in $B$ into $P$
15: output $P$
end procedure

Find-k-CPC($P$, $C$, $k$)
```

and complete-link: the similarity between one cluster and another is the averaged similarity from any member of one cluster to any member of the other cluster; it is less susceptible to outliers and elongated chains.

5 Experimental Evaluations

5.1 Data Sets

We conduct the performance evaluations based on Reuters-21578[1] corpus, which is popular for document classification evaluation purpose. It contains 21,578 documents manually grouped into 135 topic classes. The size of classes is very unbalanced, ranging from 1 to 3945. Many documents have multiple category labels, and documents in each cluster have a broad scape of contents. In our experiments, we select documents that are assigned to one or more topics, and have the attribute LEWISSPLIT="TEST" with <BODY> and </BODY> tags. There are 2,745 such original documents, denoted by OC2745, from which we then extract 2,349 documents with unique class labels to form our data set UC2349, and the rest of 396 documents with multiple classes to form MC396. Table 1 shows the statistics of these three resulted data sets.

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578

Table 1. Statistics of data sets OC2745, UC2349 and MC396 extracted from Reuter-21578 corpus

	OC2745	UC2349	MC396
# of documents	2745	2349	396
# of classes	92	58	87
max class size	1045	1041	127
min class size	1	1	1
avg. class size	38	41	13

5.2 Evaluation Metrics

We adopt two quality metrics widely used for document clustering [17], i.e., F-measure and Entropy. The F-measure of a class i is defined as $F(i) = \frac{2PR}{P+R}$. The precision and recall of a cluster j with respect to a class i are defined as: $P = Precision(i, j) = \frac{N_{ij}}{N_j}$ and $R = Recall(i, j) = \frac{N_{ij}}{N_i}$, where N_{ij} is the number of members of class i in cluster j, N_j is the size of cluster j, and N_i is the size of class i. The overall F-measure of the clustering result is the weighted average of $F(i)$:

$$F = \frac{\sum_i (|i| \times F(i))}{\sum_i |i|}$$

where $|i|$ is the number of documents in class i.

Entropy provides a measure of homogeneity of a cluster. The higher the homogeneity, the lower the entropy, and vice versa. For every cluster j in the clustering result, we compute p_{ij}, the probability that a member of cluster j belonging to class i. The entropy of each cluster j is calculated using $E_j = -\sum_i p_{ij} \log(p_{ij})$, where the sum is taken over all classes. The total entropy for a set of clusters is calculated as the sum of entropies of each cluster weighted by its size:

$$E = \sum_{j=1}^{m} (\frac{N_j}{N} \times E_j)$$

where N_j is the size of cluster j, m is the number of clusters, and N is the size of document collection.

5.3 Performance Evaluation

Experiment 1. Table 2 shows the performance of CPM given the size of threshold clique. Obviously CPM produces more clusters than the number of categories in the benchmark. This is because Reuters corpus are manually classified according to a set of pre-defined keywords (one for each class roughly). Thus the schema of categorization is rather unifarious. One document may belong to far more groups since the grouping criterion could be diverse. CPM is less limited by external constrains, which favors multifarious categorization schemes, and thus has more clusters. The least number of clusters are found at $k = 2$ where

CPM is degenerated to find connected components, which actually partitions the collections. With larger k, the cluster number increases as larger k allows for more overlapping clusters.

In terms of both F-measure and Entropy, CPM performance improves rapidly at the first few k augments, but worsens slowly with the further increases. Interestingly, there are some close optimal values of k on these data sets around 4–6. Unlike our expectation, the results on MC396, which contains documents all belonging to multiple classes, show inconsistencies on F-measure and Entropy. For Entropy, it is reasonable that CPM performs the best on MC396 since CPM favors overlapping clusters. But F-measure gives the worst results on it. F-measure seems very sensitive to outliers and penalizes their recalls heavily. As we found relatively larger proportion of outliers in MC396 clustering results, this may explain the low F-values.

We originally expected that the results on OC2745 would be far and few between UC2349 and MC396, but the worst Entropy results are observed on it. One possible reason is that Entropy favors small cluster number and cluster size. This may also explain the obviously low Entropy values on MC396 other than the advantages on overlapping clusters. Note that when $k = 2$, the performance is significantly poorer than other choices. This is also because at $k = 2$, CPM algorithm can only find connected components, which are the most relaxed criterion for clustering.

Table 2. Performance of CPM with respect to different sizes of the threshold clique

	# of clusters			F-measure			Entropy		
k	OC2745	UC2349	MC396	OC2745	UC2349	MC396	OC2745	UC2349	MC396
2	**1045**	**874**	**212**	0.093	0.083	0.234	0.596	0.366	0.749
3	1157	962	281	0.287	0.353	0.287	0.499	0.413	0.177
4	1455	1318	281	0.398	0.407	**0.302**	**0.481**	**0.376**	**0.140**
5	1964	1813	286	**0.525**	0.503	0.294	0.490	0.383	0.153
6	2390	2392	291	0.495	**0.594**	0.289	0.543	0.411	0.153
7	3177	3045	294	0.488	0.568	0.284	0.635	0.458	0.163
8	3782	3762	297	0.421	0.501	0.277	0.740	0.514	0.164
9	4503	4330	300	0.409	0.488	0.269	0.903	0.549	0.164
10	5261	4894	300	0.401	0.441	0.269	0.907	0.587	0.164

Experiment 2. In this experiment, we compare CPM with the other two representative clustering algorithms, k-means and complete-link. Because it is impossible to command CPM to produce exact number of clusters with the benchmark, we use $k = 4$, at which CPM reaches nearly optima based on Table 2. To make comparisons fair under this condition, we examine both k-means and complete-link twice: one uses the same number of clusters as the benchmark, and the other uses the same number of clusters as CPM, which are denoted by KM-B, KM-C, CL-B, and CL-C (suffixes B and C represent Benchmark and CPM, respectively). Furthermore, because k-means is well-known to be sensitive to local

optima, we repeat the algorithm 50 times with different initial centroids and average the outcomes achieved. The threshold for complete-link distance measure is set according to the computed values of t_c by CPM (see Section 3). This is to align with CPM.

Table 3 shows that CPM performs worse than k-means and complete-link if the standard number of clusters as the benchmark are produced. Because CPM generates far more clusters than the standard, this comparison is somewhat unfair to CPM. However, when the number of CPM clusters is used, its advantages can be clearly observed. Under this condition, k-means performs the worst among the three. Its poor performance on MC396 is very obvious because k-means can only produce partitioning of the corpus. Complete-link clusters are non-overlapping MDCs. The results show that CPM outperforms complete-link on all three test sets as well. This testifies the advantages of our method over the typical conventional clustering algorithms in terms of unrestraint cluster number.

Table 3. Comparisons of CPM and the other two representative algorithms, k-means (KM) and complete link (CL). We use $k = 4$.

	# of clusters			F-measure			Entropy		
	OC2745	UC2349	MC396	OC2745	UC2349	MC396	OC2745	UC2349	MC396
CPM	1455	1318	281	0.398	0.407	0.302	0.481	0.376	0.140
KM-B	92	58	87	0.510	0.503	0.391	0.319	0.270	0.117
KM-C	1455	1318	281	0.294	0.251	0.190	0.520	0.475	0.358
CL-B	92	58	87	0.531	0.511	0.437	0.276	0.203	0.105
CL-C	1455	1318	281	0.362	0.305	0.285	0.503	0.447	0.266

6 Conclusion and Future Work

We present a novel clustering algorithm CPM by applying clique percolation technique introduced from the area of biological physics. A more generalized framework related to it is the so-called "small-world network" describing many kinds of community structures in nature and society, which is extensively studied in random networks [9]. This is the pioneer work for the CPM being applied in document clustering. The preliminary results demonstrate it is feasible and promising for document clustering. We are confident that CPM is interesting and worth of further studies. There are still many issues left to be studied more deeply. So far, the heuristic relationship between p_c, N and t_c has not been well studied. To generate an appropriate random graph, an alternative is to make use of the degree distribution of graph vertices. For each vertex, some nearest neighbors associated with the precise degree distribution can be considered. This will lead to the further exploration on techniques to analyze complex networks. Furthermore, due to the NP-hardness of MDC enumeration algorithms, the CPM is time-consuming. Improvements on efficiency are required. In the future, we will also compare CPM to some more advanced clustering algorithms.

References

1. Baker, L., McCallum, A.: Distributional clustering of words for text classification. In Proc. of ACM SIGIR (1998):96–103
2. Bezdek, J.C.: Pattern recognition with fuzzy objective function algorithms. Plenum Press, New York
3. Bron, C., Kerbosch, J.: Finding all cliques of an undirected graph. Communications of the ACM **16** (1971):575–577
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein C.: Introduction to algorithms, 2nd Edition. McGraw-Hill
5. Cutting, D., Karger, D., Pedersen, J., Tukey, J.W.: Scatter/Gather: A cluster-based approach to browsing large document collections. In Proc. of the 15th ACM SIGIR Conference (1992):318–329
6. Derenyi, I., Palla, G., Vicsek T.: Clique percolation in random networks. Physics Review Letters **95** (2005):160202
7. Dhillon, I.S.: Co-clustering documents and words using bipartite spectral grpah partitioning. In Proc. of the 7th ACM-KDD (2001): 269–274
8. Ding, C.H.Q., He, X.F., Zha, H.Y., Gu, M., Simon, H.D.: A min-max cut algorithm for graph partitioning and data clustering. In Proc. of IEEE ICDM (2001) 107–114
9. Dorogovtsev, S.N., Mendes, J.F.F.: Evolution of networks. Oxford Press, New York
10. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys **31** (1999):264–323
11. King, B.: Step-wise clustering procedures. Journal of the American Statistical Association **69** (1967):86–101
12. Krishnapuram, R., Joshi, A., Nasraoui, O., Yi, L.Y.: Low-complexity fuzzy relational clustering algorithms for web mining. IEEE Transactions on Fuzzy Systems **9** (2001):595–607
13. Liu, X., Gong, Y.: Document clustering with clustering refinement and model selection capabilitities. In Proc. of ACM SIGIR (2002):191–198
14. Palla, G., Derenyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex netowrks in nature and society. Nature **435** (2005):814–818
15. Raghavan, V.V., Yu, C.T.: A comparison of the stability characteristics of some graph theoretic clustering methods. IEEE Transactions on Pattern Analysis and Machine Intelligence **3** (1981):393–402
16. Sneath, P.H.A., Sokal, R.R.: Numerical taxonomy: the principles and practice of numerical classification. Freeman, London, UK
17. Steinbach, M., Karypis, G., Kumar, V.: A comparison of doucment clustering techniques. In Proc. of KDD-2000 Workshop on Text Mining (2000)
18. Tsukiyama, S., Ide, M., Ariyoshi, H., Shirakawa, I.: A new algorithm for generating all the maximal independent sets. SIAM Journal on Computing **6** (1977):505–517
19. Zhao, Y., Karypis, G.: Criterion functions for document clustering. Technical Report #01-40, Department of Computer Science, University of Minnesota