

# Natural Document Clustering by Clique Percolation in Random Graphs

Wei Gao and Kam-Fai Wong

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

**Abstract.** Document clustering techniques mostly depend on models that impose explicit and/or implicit priori assumptions as to the number, size, disjunction characteristics of clusters, and/or the probability distribution of clustered data. As a result, the clustering effects tend to be unnatural and stray away more or less from the intrinsic grouping nature among the documents in a corpus. We propose a novel graph-theoretic technique called *Clique Percolation Clustering* (CPC). It models clustering as a process of enumerating adjacent maximal cliques in a random graph that unveils inherent structure of the underlying data, in which we unleash the commonly practiced constraints in order to discover natural overlapping clusters. Experiments show that CPC can outperform some typical algorithms on benchmark data sets, and shed light on natural document clustering.

## 1 Introduction

Document clustering is an important technique that facilitates the navigation, search and analysis of information in large unstructured text collections. It uses an unsupervised process to identify inherent groupings of similar documents as a set of clusters such that the intra-cluster similarity is maximized and the inter-cluster similarity is minimized.

Generally, clustering has three fundamental issues to solve: a data presentation model, a data similarity measure, and a clustering algorithm that builds the clusters using the data model and the similarity measure. Most existing clustering methods are based on vector space model [1,17] and represent document as a feature vector of unique content-bearing words that occur in the document sets, which is also known as “bag-of-words” model. Document similarity is calculated using one of the mathematical association measures, such as Euclidean distance, Cosine, Overlap, or Dice coefficients, etc., formulated with the feature vectors. Many clustering models and algorithms have been proposed. From different perspectives, they can be categorized into agglomerative or divisive, hard or fuzzy, deterministic or stochastic [11].

Most existing clustering algorithms optimize criterion functions with respect to the similarity measure in use over all the documents assigned to each potential partition of the collection [11,22]. They always impose some explicit and/or

implicit constraints with respect to the number, size, shape and/or disjoint characteristics of target clusters. For example, partitional algorithms like  $k$ -means assumes the cluster number  $k$  and does not allow one document belonging to multiple groups. Although fuzzy clustering, such as fuzzy  $C$ -means/medoids algorithm [3,13], does support overlapping clusters by the membership function and fuzzifier parameter, they are still confined by cluster number and can find only spherical shape clusters (due to the assumption like  $k$ -means that each cluster can be described by a spherical Gaussian). Some algorithms, e.g.  $EM$  (Expectation-Maximization) clustering, are model-based, assuming Naive Bayes or Gaussian Mixture model [2,14]. They strongly presume certain probabilistic distributions of clustered documents and try to find the model that maximizes the likelihood of the data. When data cannot fit the hypothetical distribution, poor cluster quality can result.  $k$ -way clustering or bisection algorithms [22] adapt all kinds of criterion functions, but require to specify cluster number and force clusters to be equally sized. Recently, spectral clustering [8,9] based on graph partitioning has emerged as one of the most effective clustering tools, whose criterion functions are based on max-flow/min-cut theorem [5]. However, they prohibit overlapping clusters which ought to be important for document clustering.

In this paper, we define natural document clustering as a problem of finding unknown number of overlapping document groups with varied sizes and arbitrary data distributions. We try to obtain the clustering results with these free characteristics by removing as many external restrictions as possible 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 Clustering* (CPC). The general idea is to identify adjacent maximal complete subgraphs (maximal cliques) in the document similarity graph using a threshold clique. Certain adjacent maximal cliques are then merged to form one of document clusters that can be fully explored by the threshold clique. Although CPC does introduce an explicit parameter  $k$ , which is the size of threshold clique, our algorithm can automatically settle the critical point, at which the natural clustering of the collection can be achieved. We show that CPC can outperform some representative algorithms with experiments on benchmark data.

The rest of this paper is organized as follows: Section 2 gives related work; Section 3 describes the proposed CPC method and its algorithmic implementation; Section 4 presents experiments and results; Finally, we conclude this paper.

## 2 Related Work

### 2.1 Graph-Based Document Representation

Two types of graph-based representations have been proposed for modeling documents in the context of clustering. One is the graph obtained by computing the pairwise similarities between the documents [9], and the other is obtained

by viewing the documents and the terms as a bipartite graph (co-clustering) [8]. Our work use the first model.

In general, suppose  $V = \{d_1, d_2, \dots, d_{|V|}\}$  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

$$M_{ij} = \begin{cases} w_{ij} & \text{if there is an edge } \{i, j\} \\ 0 & \text{otherwise} \end{cases}, \quad (1)$$

where each entry  $M_{ij}$  is the edge weight, and  $w_{ij}$  is the value of similarity metric (in what follows we assume Cosine coefficient) between  $d_i$  and  $d_j$ . The graph can also be unweighted where an edge exists indicating that the distance of its two vertices is smaller than some threshold, in which case  $M_{ij}$  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. The maximal cliques are considered the strictest definition of a cluster [16]. In graph theory, enumerating all maximal cliques (equivalently, all maximal independent sets or all minimal vertex covers) is a fundamental combinatorial optimization problem and its worst-case complexity is believed NP-hard [4,21].

## 2.2 Graph-Theoretic Clustering

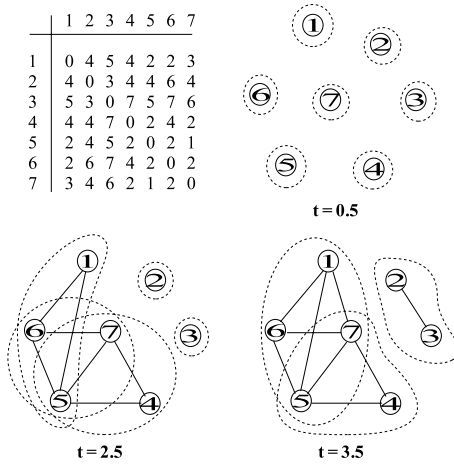
Traditional hierarchical agglomerative clustering (HAC) are intrinsically graph-based. HAC treats each data point as a singleton cluster and then successively merges pairs of clusters until all clusters have been merged into a single cluster that contains all documents. Single-link, complete-link and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

In single-link algorithm [19], 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 [12] 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 [6] is a compromise between single-link 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.

### 3 Clique Percolation Clustering Model

#### 3.1 Preliminaries

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  $t$ . Fig. 1 illustrates that given a 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 maximal clique were considered as a cohesive form of cluster, we could 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. Different from HAC clusters, they are planar and overlapping rather than hierarchical and disjoint. They also display interesting properties of natural clusters except for excessive intra-cluster cohesiveness like complete-link clusters.



**Fig. 1.** Graphs with respect to the threshold level  $t$  and different cohesive clusters (in dotted regions) resulted from the respective  $t$  value

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 some probability  $p$ , the probability that 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 a monotonic manner. In order for an appropriate  $t$ , we first try to determine  $p_c$ , the critical value of  $p$ , and then derive  $t$  given  $p_c$  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 [7].

At this threshold, the percolation transition takes place (see Section 3.2). For clustering, the assumption behind is that no cluster can be excessively larger than others by commanding  $p < p_c$ .

### 3.2 $k$ -Clique Percolation

**Concepts.** The concept of  $k$ -clique percolation for random networks was recently studied in biological physics in [7]. Its successful applications for uncovering community structure of co-authorship networks, protein networks and word association graphs can be found in [15]. Here we briefly describe some related notions.

**Definition 1.**  $k$ -clique is a complete subgraphs of  $k$  vertices.

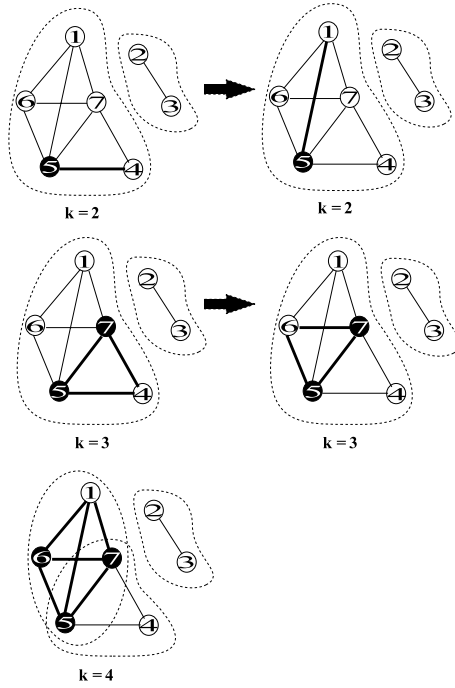
**Definition 2.**  $k$ -clique adjacency: Two  $k$ -cliques are adjacent if they share  $k - 1$  vertices, i.e., if they differ only in a single vertex.

**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.

**Definition 4.**  $k$ -clique adjacency graph is a compressed transformation 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 from one vertex of a  $k$ -clique adjacency graph to another along an edge is equivalent to rolling a  $k$ -clique template (threshold clique) from one  $k$ -clique of the original graph to an adjacent one. A  $k$ -clique template can be thought of as an object that is isomorphic to a complete graph of  $k$  vertices. It 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 its other  $k - 1$  vertices fixed. Thus, the  $k$ -clique percolation clusters of a graph are all those subgraphs that can be fully explored by rolling a  $k$ -clique template in them [7]. Fig. 2 illustrates the effects of one-step rolling of a  $k$ -clique template (for  $k = 2, 3, 4$  when  $t = 3.5$ ) that produce different topologies of clusters. Note that a  $k$ -clique percolation cluster is equivalent with all maximal cliques adjacent by at least  $k - 1$  vertices. Thus, compared to the strict maximal clique clusters aforementioned (see Section 3.1), the cohesiveness of a  $k$ -clique percolation cluster can be adjusted by the  $k$  value. In addition, such clusters are connected components on a  $k$ -clique adjacency graph that can be discovered with efficient algorithms. The goal of CPC 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  $k$ ? Under such  $p_c$  (critical point), a giant  $k$ -clique percolation cluster that is excessively larger than other clusters will take place [10,7]. Intuitively, the greater the value of  $p$  ( $p > p_c$ ), the more likely the giant cluster appears, and the bigger its size is (which includes most graph nodes), as if using a  $k$ -clique can percolate the entire graph.



**Fig. 2.** Effects of  $k$ -clique template rolling in a relocation step (black nodes are fixed and bold edges are involved when rolling) with respect to different  $k$  that results in different  $k$ -clique percolation clusters (in dotted regions)

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  $p_c^{k-1}$  is acceptable, because each of the new  $k-1$  edges must exist in order to reach a new  $k$ -clique after relocation. Therefore, we get the percolation threshold function  $p_c(k)$  with respect to  $k$  and  $|V|$ :

$$p_c(k) = \frac{1}{[(k-1)(|V|-k)]^{\frac{1}{k-1}}} . \tag{2}$$

For  $k = 2$  in particular,  $p_c(2) = 1/(|V|-2)$  gives the percolation threshold of 2-clique connectedness (edge connectedness) of the graph, i.e., most graph nodes can be fully explored by a traversal along the edges.

**Generation of Random Graph.** By no means, an appropriate graph for clustering can be obtained without prior information regarding the global or local statistics of node connectivity in terms of certain degree distribution. In order to generate such a graph, one commonly specifies a series of hard threshold values of edge weight  $t$ , and then determines a good value  $t_c$  by trial and error. However, its time cost is generally very expensive due to the complexity of graph-theoretic approaches. Thus,  $t_c$  is usually hard to achieve. The concept of clique percolation provides a fundamental probabilistic formalism for determining the critical point, with which we can estimate  $t_c$  more directly without prior knowledge on statistics of graph and save the time cost of trial and error.

We examine the co-relation between  $p$  and  $t$ . Given  $p_c$ , we estimate the bound(s) of  $t_c$  so that the graph with the equivalent connectivity as that at the percolation threshold can be generated. Because  $p$ - $t$  are monotone, an appropriate graph for clustering could be obtained using  $t$  slightly less than  $t_c$ . This is to prohibit the emergence of a giant cluster at the critical point. For simplification, we derive the upper bound of  $t_c$  by an approximation:

$$t_c(k) = p_c(k) \times (w_{max} - w_{min}) \quad , \quad (3)$$

where  $w_{max}$  and  $w_{min}$  are the maximum and minimum values of document similarity in the collection, respectively.

### 3.3 Algorithmic Implementation

The clustering process is turned out to be a problem of finding all maximal cliques and then merging those with  $k - 1$  common nodes into clusters. The proposed CPC method includes 5 major steps:

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  $p_c(k)$  using (2), and compute  $t_c(k)$  using (3).
3. For each entry in matrix  $\mathbf{A}$ , if  $w_{ij} < t_c(k)$ , then reassign  $w_{ij} = 1$ , otherwise set  $w_{ij} = 0$ ; Create document similarity graph  $G$  using the updated  $\mathbf{A}$  as adjacent matrix.
4. Enumerate all maximal cliques in  $G$  using Algorithm 1.
5. Create a  $M \times M$  adjacent matrix  $\mathbf{B}$  (where  $M$  is the number of maximal cliques identified), find  $k$ -clique percolation clusters using Algorithm 2 on  $\mathbf{B}$ .

**Enumerating Maximal Cliques.** Algorithms for finding maximal cliques (step 4 above) was studied in [4,21] and achieved processing time bounded by  $O(v^2)$  and  $O(nm\mu)^1$ , respectively. Their algorithms are distinctive because they

<sup>1</sup>  $v$  is the number of maximal cliques.  $n$ ,  $m$  and  $\mu$  are the number of the vertices, edges and all the maximal independent sets, respectively. Note that each maximal independent set of a graph  $G$  corresponds one-to-one to each maximal clique of the complementary graph of  $G$  [4,21].

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 maximal clique is output at each end of back-track. Thus the running time is  $O(v)$ . Because  $v$  may be exponential with the growth of the number of vertices in worst case, our method is not a polynomial time algorithm either.

---

**Algorithm 1.** Enumerate All Maximal Cliques

---

**input:** Vertex set  $V$  and edge set  $E$  of graph  $G$ .

**output:** All maximal cliques of  $G$  into  $MC$ .

**procedure** EnumMC ( $MC, U, E$ )

```

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

```

**end procedure**

$MC \leftarrow \phi$

EnumMC ( $MC, V, E$ )

---

**Finding  $k$ -Clique Percolation Clusters.** When all the maximal cliques are enumerated, a clique-clique adjacent matrix is prepared. It is symmetric where each row (column) represents a maximal clique and matrix values are equal to the number of common vertices between the two cliques (the diagonal entries are the clique sizes). Note that the intersection of two cliques is always a clique with at least  $k - 1$  (common) nodes. The  $k$ -clique percolation clusters are the one-to-one correspondents to the connected components in the clique-clique adjacency graph, which can be obtained using Algorithm 2 (step 5 above). The algorithm first creates a clique-clique adjacent matrix  $\mathbf{B}$ , 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. The resulted connected components are used as indices of maximal cliques for outputting  $k$ -clique percolation clusters.

## 4 Experimental Evaluations

### 4.1 Data Sets

We conduct the performance evaluations based on Reuters-21578<sup>2</sup> corpus, which is popular for document clustering purpose. It contains 21,578 documents that

<sup>2</sup> <http://www.daviddlewis.com/resources/testcollections/reuters21578>

**Algorithm 2.** Find All  $k$ -Clique Percolation Clusters**input:** A set of all maximal cliques  $MC$  and  $k$ .**output:** All  $k$ -clique percolation clusters into  $CPC$ .**procedure** Find-k-CPC ( $CPC, MC, k$ )1:  $B \leftarrow 0$  {Initialize entries in  $B$  as 0}2: **for**  $i$  from 1 to  $M$  **do**3:   **for**  $j$  from 1 to  $M$  **do**4:      $B[i][j] \leftarrow |MC_i \cap MC_j|$  {Count common nodes of two maximal cliques}5:     **if**  $(i = j) \wedge (B[i][j] < k)$  **then**6:        $B[i][j] \leftarrow 0$  {Off-diagonal element  $< k$  is replaced by 0}7:     **else if**  $(i \neq j) \wedge (B[i][j] < k - 1)$  **then**8:        $B[i][j] \leftarrow 0$  {Diagonal element  $< k - 1$  replaced by 0}9:     **end if**10:   **end for**11: **end for**12:  $CC \leftarrow \phi$  {Initialize connected component set  $CC$ }13:  $i \leftarrow 1$  {Initialize recursion counter  $i$ }14:  $CC \leftarrow \text{DFS}(CC, B, i)$  {Recursive DFS to find connected components  $CC$  in  $B$ }15:  $CPC \leftarrow \text{OutputCPC}(CC, MC)$  { $CC$  is as index of  $MC$  for the output of  $CPC$ }**end procedure**Find-k-CPC ( $CPC, MC, k$ )

are manually grouped into 135 classes. The number of documents for different clusters is very unbalanced, ranging from 1 to 3,945. Many documents have multiple category labels, and documents in each cluster have a broad scope of contents. In our experiments, we remove the clusters with less than 5 documents. We then extract 9,459 documents with unique class labels to form one of our data sets TS1, and rest of 11,084 documents with multiple class labels form TS2. At last, we result in 51 classes in TS1 and 73 classes in TS2. Table 1 shows the statistics of the original Reuters corpus (ORG) and the two resulted data sets.

## 4.2 Evaluation Metrics

We adopt two quality metrics widely used for document clustering [20], 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:  $P = \text{Precision}(i, j) = \frac{N_{ij}}{N_j}$  and  $R = \text{Recall}(i, j) = \frac{N_{ij}}{N_i}$ , 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  $F(i)$ :

$$F = \frac{\sum_i (|i| \times F(i))}{\sum_i |i|}, \quad (4)$$

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

**Table 1.** Statistics of data sets ORG (Reuters-21578 original corpus), TS1 and TS2

	ORG	TS1	TS2
# of documents	21578	9459	11084
# of clusters	135	51	73
max cluster size	3945	3945	3508
min cluster size	1	5	5
avg. cluster size	186	153	167

Entropy provides a measure of homogeneity of a cluster. The higher the homogeneity of a cluster, 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  $j$  belongs 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 the size of that cluster:

$$E = \sum_{j=1}^m \left( \frac{N_j}{N} \times E_j \right), \quad (5)$$

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

### 4.3 Performance Evaluation

**Experiment 1.** Table 2 shows the performance of CPC given the different size of threshold clique. Obviously CPC 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). Thus the schema for the categorization is rather unifarious. One document with less discriminative features may belong to more groups and the grouping criterion could be more diverse. CPC is less limited by external constraints, which favors multifarious categorization schemes, and thus has more clusters.

The results on TS2 are better than on the other two data sets in terms of both F-measure and Entropy. Because TS2 contains documents all belonging to multiple classes, we think the better results on it can be attributed to CPC favoring overlapping clusters. We originally expected that the results on ORG would be far and few between the performances on TS1 and TS2, but the worst results of the three are observed on it. One possible reason is that we had pruned the classes with less than 5 documents for producing TS1 and TS2, where fewer small clusters are remained. This may indicate that CPC is disadvantageous in identifying excessively small clusters. We can also observe that the algorithm gives the best results when  $k = 4$ . Note that when  $k = 2$ , the performance is significantly poorer than other choices of  $k$ . This is because at  $k = 2$ , the procedure of CPC algorithm is degenerated to find connected components, which is regarded as the most relaxed criterion for clustering.

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

$k$	# of clusters			F-measure			Entropy		
	ORG	TS1	TS2	ORG	TS1	TS2	ORG	TS1	TS2
2	454	106	183	0.501	0.529	0.542	0.381	0.319	0.322
3	328	94	112	0.762	0.836	0.863	0.224	0.107	0.109
4	273	85	97	0.748	0.833	0.875	0.205	0.111	0.104
5	306	91	124	0.603	0.771	0.852	0.237	0.135	0.107
6	362	99	138	0.579	0.725	0.852	0.253	0.151	0.107

**Experiment 2.** In this experiment, we compare CPC with the other three representative clustering algorithms,  $k$ -means, single-link and complete-link. Because it is impossible to command CPC to produce exact number of clusters with the benchmark, we use  $k = 4$  for the threshold clique, which is the optimal solution based on Table 2 and also brings about the closest number of clusters to the benchmark. To make fair comparisons under this condition, we examine every one of  $k$ -means, single-link and complete-link twice: one with the same number of clusters as the benchmark, and the other with the same number of clusters as CPC. The results are denoted by KM-b, KM-c, SL-b, SL-c, CL-b and CL-c (suffixes b and c represent benchmark and CPC, respectively). Furthermore, because  $k$ -means is well-known to be sensitive to local optima, we repeat the algorithm 50 times with different initializations (initial centroids) and choose the best outcomes achieved. In order to align with the clustering results of CPC and the benchmark, we stop the HAC process of single-link and complete-link when the specified number of clusters are left.

Table 3 shows that CPC outperforms other algorithms on all three test sets irrespective of the cluster number used. When compared with KM-b, SL-b and CL-b, CPC can only produce proximate number of clusters, but performs better on both measurements. This indicates that CPC clustering, although multifarious, is still more accurate than other clusterings with exact the same number of clusters as benchmark. When using the same number of clusters as CPC, the results of KM-c, SL-c and CL-c are even worse in some extent. In addition, CPC performs better on TS2 than TS1, and all the remaining algorithms demonstrate an opposite outcome, i.e., the results on TS1 are relatively better than TS2. This testifies the advantages of our method over the partitional algorithms that can only produce disjoint clusters.

$k$ -means performs the worst among the three. Its poor performance on TS2 is very obvious because  $k$ -means can only produce spherical partitional clusters of the corpus. Single-link results in clusters that are connected components and complete-link clusterings are non-overlapping cliques. It is reasonable for complete-link performing better than single-link. The superiority of CPC stems from several main reasons: First, CPC aims to form natural clusters that should be by all means overlapping for Reuters corpus; Second, the cohesiveness of CPC clusters is moderate in comparison with relaxed single-link and restricted

**Table 3.** Comparisons of CPC and the other three representative algorithms,  $k$ -means (KM), single-link (SL) and complete-link (CL). We use  $k = 4$ 

	# of clusters			F-measure			Entropy		
	ORG	TS1	TS2	ORG	TS1	TS2	ORG	TS1	TS2
CPC	273	85	97	0.748	0.833	0.875	0.205	0.111	0.104
KM-b	135	51	73	0.512	0.631	0.391	0.310	0.286	0.372
KM-c	273	85	97	0.503	0.509	0.344	0.326	0.315	0.360
SL-b	135	51	73	0.547	0.572	0.466	0.307	0.290	0.325
SL-c	273	85	97	0.533	0.559	0.485	0.310	0.302	0.317
CL-b	135	51	73	0.694	0.727	0.715	0.219	0.279	0.195
CL-c	273	85	97	0.670	0.759	0.734	0.235	0.162	0.183

complete-link; Third, because CPC does not assume each cluster described by any distribution, it tends to be more flexible and natural than model-based approaches like spherical  $k$ -means.

## 5 Conclusion and Future Work

We present a novel clustering algorithm CPC 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 [10]. This is the first time for the clique percolation being applied in document clustering. The preliminary results demonstrate it is feasible and promising for document clustering. We are confident that CPC is interesting and worth of further studies. There are still many issues left to be studied more deeply. One is the determination of threshold values of  $t_c$  according to the percolation threshold probability  $p_c$ . So far, the mathematical relationship between them is not exact and clear-cut. To generate an appropriate random graph from  $p_c$ , an alternative is to make use of the degree distribution of graph vertices. For each vertex, some nearest neighbors associated with its degree distribution are considered to produce the connectivity instead of depending on the harsh cut by  $t_c$  (derived from  $p_c$ ). This will lead to the further exploration on techniques to analyze complex networks. Furthermore, due to the NP-hardness of maximal clique enumeration algorithms, the CPC method is time-consuming. More efficient maximal cliques enumeration algorithm is required. In the future, we will also compare CPC to more advanced clustering algorithms, such as spectral clustering [8,9] and Information Bottleneck method [18].

## References

1. Baeza-Yates, R., Ribeiro-Neto, B.: Modern information retrieval. Addison-Wesley, New York
2. Baker, L., McCallum, A.: Distributional clustering of words for text classification. In Proc. of the 21th ACM SIGIR Conference (1998):96–103

3. Bezdek, J.C.: Pattern recognition with fuzzy objective function algorithms. Plenum Press, New York
4. Bron, C., Kerbosch, J.: Finding all cliques of an undirected graph. *Communications of the ACM* **16** (1971):575–577
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein C.: Introduction to algorithms, 2nd Edition. McGraw-Hill
6. 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
7. Derenyi, I., Palla, G., Vicsek T.: Clique percolation in random networks. *Physics Review Letters* **95** (2005):160202
8. Dhillon, I.S.: Co-clustering documents and words using bipartite spectral graph partitioning. In Proc. of the 7th ACM-KDD (2001):269–274
9. 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
10. Dorogovtsev, S.N., Mendes, J.F.F.: Evolution of networks. Oxford Press, New York
11. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. *ACM Computing Surveys* **31** (1999):264–323
12. King, B.: Step-wise clustering procedures. *Journal of the American Statistical Association* **69** (1967):86–101
13. 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
14. Liu, X., Gong, Y.: Document clustering with clustering refinement and model selection capabilities. In Proc. of the 25th ACM SIGIR Conference (2002):191–198
15. Palla, G., Derenyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435** (2005):814–818
16. 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
17. Salton, G.: Automatic text processing: the transformation, analysis, and retrieval of information by computer. Addison-Wesley, New York
18. Slonim, N., Tishby, N.: Document clustering using word clusters via the information bottleneck method. In Proc. of the 23th ACM SIGIR Conference (2000): 208–215
19. Sneath, P.H.A., Sokal, R.R.: Numerical taxonomy: the principles and practice of numerical classification. Freeman, London, UK
20. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In Proc. of KDD-2000 Workshop on Text Mining (2000)
21. 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
22. Zhao, Y., Karypis, G.: Criterion functions for document clustering. Technical Report #01-40, Department of Computer Science, University of Minnesota