Extending Consensus Clustering to Explore Multiple Clustering Views

Yi Zhang, Tao Li
School of Computer Science
Florida International University
Miami, FL 33199, USA
Email: {yzhan004,taoli}@cs.fiu.edu

Abstract

Consensus clustering has emerged as an important extension of the classical clustering problem. Given a set of input clusterings of a given dataset, consensus clustering aims to find a single final clustering which is a better fit in some sense than the existing clusterings. There is a significant drawback in generating a single consensus clustering since different input clusterings could differ significantly. In this paper, we develop a new framework, called Multiple Consensus Clustering (MCC), to explore multiple clustering views of a given dataset from a set of input clusterings. Instead of generating a single consensus, MCC organizes the different input clusterings into a hierarchical tree structure and allows for interactive exploration of multiple clustering solutions. A dynamic programming algorithm is proposed to obtain a flat partition from the hierarchical tree using the modularity measure. Multiple consensuses are finally obtained by applying consensus clustering algorithms to each cluster of the partition. Extensive experimental results on 11 real world data sets and a case study on a Protein-Protein Interaction (PPI) data set demonstrate the effectiveness of our proposed method.

1 Introduction

Consensus clustering, also called as aggregation of clusterings (or partitions) or ensemble clustering, has emerged as an important elaboration of the classical clustering problem [1][2]. It refers to the problem of finding a single (consensus) clustering from a number of different (input or base) clusterings (or partitions) that have been obtained for a particular dataset. Many different approaches have been developed recently to solve consensus clustering problem [1][2][3][4][5]. More recently, several approaches have also been proposed to select a subset of input clusterings to form a smaller but better performing cluster consensus than using all available solutions [6][7]. Typically, in these current consensus clustering approaches, all the input clustering solutions or the selected subset of input clustering solutions are combined together to output a ***single*** consensus clustering of the data that is "better" than the existing clusterings, i.e., in this consensus clustering, clusters are better separated, or equivalently, the clustering objective functions are improved.

There is, however, a significant drawback in generating a single consensus clustering. Recent studies have shown that in consensus clustering: 1) different input clusterings could differ significantly, and 2) subsets of input clusterings could be highly correlated [3][7][8]. When different input clusterings differ significantly, the consensus by simply averaging is really a brute-force voting and there is no real "consensus" in their original meaning. As a result, a single "consensus" may not be ideal in many cases and finding a single consensus clustering solution is not always the best way to explore hidden pattern structures for a given dataset [8].

Real world datasets such as text and biology datasets are often multi-faceted with high dimensions. They can often be interpreted in many different ways and can have different clusterings that are reasonable and interesting from different perspectives [8][9]. In fact, in many datasets, clusters overlap substantially and natural clusters cannot be defined clearly. In general, a single (even the "best" if exists) clustering objective function can not effectively model the vast different types of datasets [10]. Therefore, it is interesting to explore *multiple clustering views* of a given data set. In addition, when the input clusterings differ significantly and constitute different groups, it is quite likely that the consensus formed by a certain group of input clusterings achieves better clustering performance than the consensus formed using all the input clusterings.

In this paper, we develop a new framework, called ***Multiple Consensus Clustering (MCC)***, to explore multiple clustering views of a given dataset from a set of input clusterings. Given a number of different (input) clusterings that have been obtained for a particular dataset, instead of generating a single consensus,

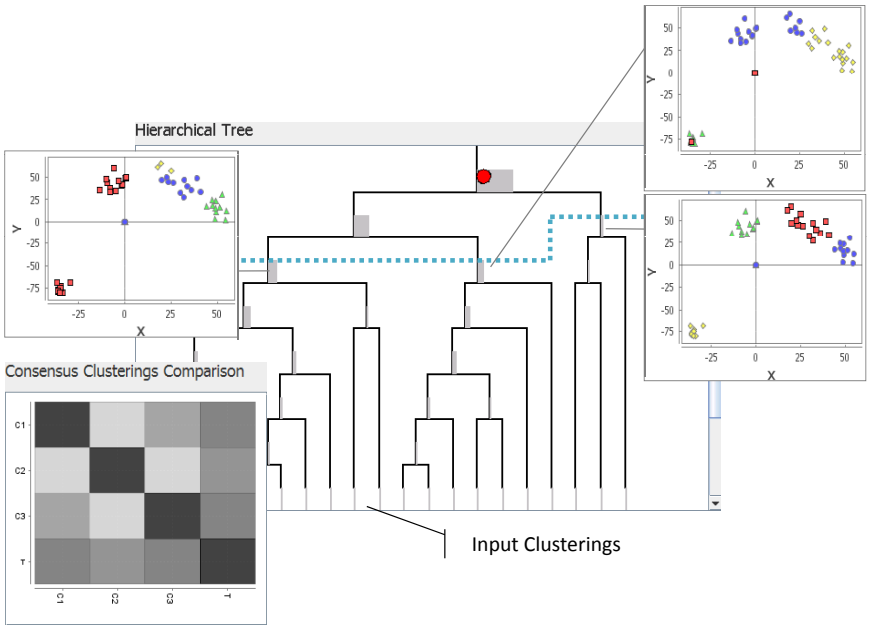

Figure 1: A screen shot of MCC.

MCC first computes the pairwise similarities between input clusterings using a Mallows distance based measure, and organizes the different input clusterings into a hierarchical tree structure using the bottom-up (agglomerative) approach. The tree structure provides an overall picture of the input clusterings and offers an informative view on their relationships. MCC provides two ways for exploring multiple clustering views. On one hand, a dynamic programming algorithm is used to automatically obtain an optimal modularity partition of clusterings from the hierarchical tree. Then for each group of the optimal partition, the consensus clustering algorithm is applied to obtain a single consensus for the input clusterings in the group. On the other hand, note that the hierarchical tree structure helps users glean insights of the cluster patterns and allows for interactive exploration of multiple clustering solutions. Instead of using the partition provided by MCC, users could cut the hierarchical tree at any level and/or select any subset of the input clusterings to explore. A screen shot of developed MCC toolkit is shown in Figure 1. In Figure 1, three clusters of input clusterings are obtained by cutting the hierarchical tree. Hence three clustering views are obtained by applying consensus clustering for each cluster of the input clusterings and the 2-D plots of the three clustering views are shown in the figure. In addition, the comparisons of different clustering views are also presented in our toolkit. Different from meta clustering that finds many alternate good clusterings of the data, our MCC generates consensus clusterings from the input clusterings of a given data set. Different from consensus clustering which finds a single consensus from the input clusterings, our MCC groups the input clusterings and obtains multiple consensuses (a consensus for each group). In summary, our method MCC brings together two interrelated but distinct themes from clustering: consensus clustering and hierarchical clustering. Given a set of input clusterings of a particular data set, it first employs hierarchical clustering to cluster the input clusterings and then uses consensus clustering to generate a consensus for each cluster of the input clusterings. Extensive experimental results on 11 real world data sets and a case study on a Protein-Protein Interaction (PPI) data set demonstrate the effectiveness of our proposed method.

The rest of the paper is organized as follows: Section 2 presents an overview of our proposed MCC framework; Section 3 introduces the Mallows distance based approach to compute the pairwise similarities between the input clusterings; Section 4 describes the hierarchical tree construction and a procedure of automatically obtaining a flat partition based on the hierarchical tree using the modularity measure; Section 5 introduces the consensus clustering algorithms used in our MCC frame-

work; Section 6 shows the experimental results on 11 real world data sets; Section 7 presents a case study on exploring multiple clustering views on a Protein-Protein Interaction (PPI) data set; Section 9 discusses related work, and finally Section 10 concludes the paper.

2 An Overview of MCC

The general framework of MCC is shown in Figure 2. The framework consists of the following 4 steps:

1. *Input Clusterings Generation* where different input (or base) clusterings are obtained possibly by different clustering algorithms with different parameters on different subspaces from the original data set.

2. *Comparing Input Clusterings* where the pairwise similarity matrix of the input clusterings is calculated. (See Section 3.)

3. *Hierarchical Tree Generation* where a hierarchical tree of the input clusterings is constructed based on the similarity matrix. In addition, a flat partition of the input clustering can be obtained by cutting the hierarchical tree. (See Section 4.)

4. *Consensus Generation* where multiple consensuses can be generated by applying consensus clustering algorithms to different groups of the flat partition. (See Section 5.)

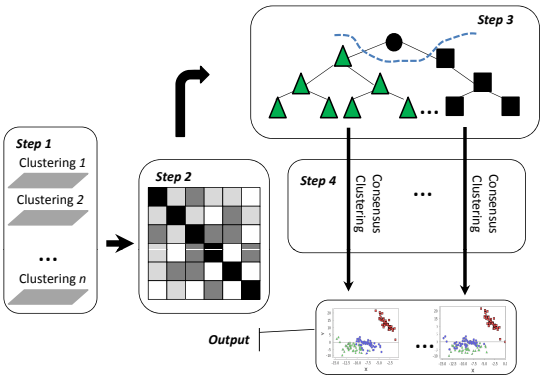

Figure 2: The general framework of MCC.

3 Comparing Input Clusterings

In order to build the hierarchical tree for the input clusterings, the similarities between different clusterings need to be computed first. Several measures can be used to compare different clusterings, such as pair counting [11], set matching [12], and variation of information [13]. One limitation of these existing measures is that they do not explicitly provide the correspondence between the clusters from different clusterings and only output the similarity measurements between different clusterings. In our work, the Mallows distance based approach is developed for comparing different input clusterings. The Mallows distance-based approach is able to compute the distance between the clusterings and also to establish the correspondence between the clusters from the different clusterings. In fact, Mallows distance provides a complete and globally optimal matching scheme between clusters in two clustering results [14][15].

Given a dataset $X = \{x_1, x_2, \ldots, x_N\}$ and two clustering results, $C1$ with J clusters and $C2$ with K clusters. Let $C1_{i,j}$ denote the probability of the i-th object belonging to the j-th cluster in $C1$, while $C2_{i,k}$ denote the probability of the i-th object belonging to the k-th cluster in $C2$. The two clusters: $C1_j$ (the j-th cluster of $C1$) and $C2_k$ (the k-th cluster of $C2$) can be represented as two vectors: $(C1_{1,j}, C1_{2,j}, \ldots, C1_{N,j})^T$, and $(C2_{1,k}, C2_{2,k}, \ldots, C2_{N,k})^T$ which are determined by the probabilities of each object x_i belonging to them, respectively. Then we assign weights α_j to $C1_j$, and β_k to $C2_k$. Note that the weights reflect the significance of the clusters and $\Sigma_{j=1}^{J}\alpha_j = 1$, $\Sigma_{k=1}^{K}\beta_k = 1$.

Thus by adopting the Mallows distance, the distance between $C1$ and $C2$ can be described as [15]:

(3.1)
$$D(C1, C2) = min_{\omega_{j,k}} \Sigma_{j=1}^{J}\Sigma_{k=1}^{K}\omega_{j,k}\Sigma_{i=1}^{N}|C1_{i,j} - C2_{i,k}|,$$

where $\omega_{j,k} \geq 0$, $\sum_{j=1}^{J}\omega_{j,k} = \beta_k$, $\sum_{k=1}^{K}\omega_{j,k} = \alpha_j$. The matching weight between $C1_j$ and $C2_k$ is indicated by $\omega_{j,k}$ and it can be computed using linear programming.

Note that the pairwise similarities are used for hierarchical clustering. Hence, based on Equation 3.1, we compute the similarity between input clusterings as follows:

$$S(C1, C2) = 1 - \frac{D(C1, C2) - minD}{maxD - minD}, \tag{3.2}$$

where $minD$ is the minimum Mallow distance between all partition pairs while $maxD$ is the maximum distance among all partition pairs.

4 Hierarchical Tree Construction

After obtaining the pairwise similarities between input clusterings, a single-link hierarchical clustering algorithm is then used to build a hierarchical tree structure using the bottom-up (agglomerative) approach. The tree structure provides an overall picture of the input clusterings and offers an informative view on their relationships. The root of the hierarchical tree T, denoted by $Root(T)$, consists of a single cluster containing all

input clusterings and the leaves correspond to individual input clusterings. Each tree node induces a cluster of input clusterings. A node-cut of the hierarchical tree is a subset C of tree nodes such that: 1) for any pair of nodes $x, y \in C$, neither x nor y is an ancestor of the other; 2) the path from every leaf of T to $Root(T)$ passes through exactly one node in C. Note that every node-cut C of the hierarchical tree induces a partition CP: each node $x \in C$ leads to one cluster that contains the input clustering corresponding to the leaves in the subtree rooted at x. Figure 3 shows an example of the hierarchical tree where the dotted line represents the node-cut recommended by our MCC by maximizing the modularity performance measure (we will described this in detail below). Note that the hierarchical tree also allows for interactive exploration of multiple clustering solution and users can cut the hierarchical tree at any level and/or select any subset of the input clusterings to explore. As shown in Figure 3, the red dot indicates the subtree that users select, and the solid line is an example of a node-cut that users define manually. Given

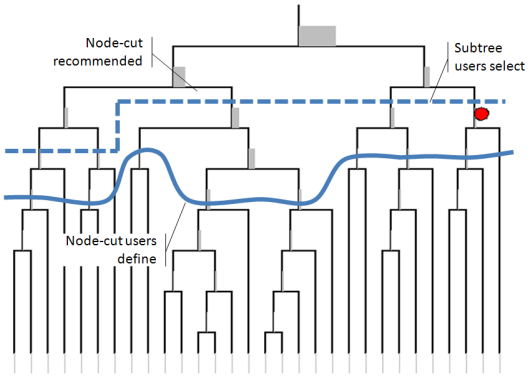

Figure 3: The hierarchical tree structure.

a hierarchical tree, there exists a large number of possible node-cuts corresponding to different partitions of the input clusterings. In order to explore multiple clustering views, the hierarchical tree needs to be converted to flat partitions. This is typically done by choosing a node-cut where each node yields a cluster consisting of all the input clusterings in the subtree rooted at that node [16][17].

We also develop an algorithm to automatically find the node-cut using the modularity. Modularity is a recently introduced clustering quality measure and has been successfully applied in many applications [18][19]. Note that each tree node x induces a cluster. Thus we can define the modularity $m(x)$ for x as follows:

$$m(x) = \frac{1}{m} \sum_{i,j \in Leaves(x)} \{A_{i,j} - \frac{k_i k_j}{2m}\}, \tag{4.3}$$

where A is the pairwise similarity matrix calculated based on Equation 3.2, k_i and k_j are the degree of node i and j which are calculated by sum of the weights of the edges connected to node i and node j, and m is the sum of the edges' weights in the hierarchical tree [20]. Given a node-cut C, its modularity $m(C)$ can then be computed as

$$m(C) = \sum_{x \in C} m(x). \tag{4.4}$$

After defining the modularity, we can find the node-cut which has the largest modularity; in other words, the goal is to find C:

$$\arg\max_C m(C) = \arg\max_C \sum_{x \in C} m(x). \tag{4.5}$$

The problem of finding the node-cut with the highest modularity can be solved efficiently using dynamic programming [21]. The problem exhibits optimal substructures as the modularity for a node is decomposable so that the total modularity is equal to the sum of the contribution of its children. The recurrence definition of the best solution for a tree node x can be expressed as

$$m^*(x) = \max\{m(x), m^*(x_1) + m^*(x_2) +, \ldots, +m^*(x_n)\}, \tag{4.6}$$

where x_1 to x_n are the node x's children from the left to the right. Using backtracking, we can find the tree nodes of the node-cut and produce a partition of the input clusterings that has the highest modularity.

5 Consensus Generation

After obtaining the partition of the input clusterings, a consensus clustering algorithm can be applied to obtain a single consensus for the input clusterings in the same cluster of the partition. Here we briefly describe the consensus clustering algorithms used in our MCC framework.

1. **PCA-based consensus algorithm**: This algorithm firstly performs dimensionality reduction on the input clusterings using Principal Component Analysis (PCA), then applies the recursive bisection algorithm to do consensus clustering [22].

2. **CSPA** (cluster-based similarity partitioning algorithm): A clustering signifies a relationship between objects in the same cluster and can thus be used to establish a measure of pairwise similarity. This induced similarity measure is then used to recluster the objects, yielding a combined clustering [23].

3. **HGPA** (HyperGraph partition algorithm): This algorithm approximates the maximum mutual information objective with a constrained minimum cut objective. Essentially, the cluster consensus problem is posed as a partitioning problem of a suitably defined hypergraph where hyperedges represent clusters [24].

4. **WClustering** (Weighted ensemble clustering) [25]: In this algorithm, each input clustering is weighted and the weights are determined in such way that the final consensus clustering provides a better quality solution.

6 Experiments

6.1 Dataset Description Firstly, we describe the data sets used in our experiments. The characteristics of the data sets are summarized in Table 1. The number of classes ranges from 2 to 10, the number of samples ranges from 47 to 4199, and the number of dimensions ranges from 4 to 1000. Further details are described below:

Table 1: Descriptions of the real-world datasets.

Datasets	# Sample	# Dim	# Class
Glass	214	9	7
Ionosphere	351	34	2
Iris	150	4	3
Soybean	47	35	4
Wine	178	13	3
Zoo	101	18	7
CSTR	475	1000	4
Log	1367	200	8
LetterIJL	227	16	3
Reuters	2900	1000	10
WebKB4	4199	1000	4

1. Seven datasets (Glass, Ionosphere, Iris, Soybean, Wine, Zoo, LetterIJL) are from UCI data repository [26]. LetterIJL is a randomly sampled subset of three {I,J,L} from Letters dataset.

2. Four datasets (CSTR, Log, Reuters, WebKB4) are standard text datasets that are often used as benchmarks for document clustering. The document datasets are pre-processed (removing the stop words and unnecessary tags and headers) using the rainbow package [27].

3. CSTR is the dataset of the abstracts of technical reports published in Computer Science departments between 1991 and 2002. The dataset contains 476 abstracts, which are divided into four research areas: Natural Language Processing(NLP), Robotics/Vision, Systems, and Theory.

4. The Log dataset contains 1367 text messages of system log from different desktop machines describing the status of computer components. These messages are divided into 8 different situations.

5. The Reuters dataset is a subset of the Reuters 21578 Text Categorization Test collection containing the 10 most frequent categories among the 135 topics.

6. The WebKB dataset contains webpages gathered from university computer science departments. There are about 8280 documents and they are divided into 7 categories: student, faculty, staff, course, project, department and other. The WebKB4 dataset is the subset of WebKB associating with four most populous entity-representing categories, i.e., student, faculty, course and project [28].

6.2 Experimental Setup All the above datasets come with labels. Viewing these labels as indicative of a reasonable clustering, we define the following accuracy measure [29]:

$$Accuracy = \max\Big(\sum_{C_k, L_m} T(C_k, L_m)\Big)/n, \tag{6.7}$$

where n is the number of data points, C_k denotes the k-th cluster, L_m is the m-th class and $T(C_k, L_m)$ is the number of data points that belong to class m and are assigned to cluster k. Accuracy is thus computed as the maximum sum of $T(C_k, L_m)$ for all pairs of clusters and classes, and these pairs have no overlap.

In the experiments, the base clusterings are obtained by running K-means 30 times.

6.3 Measuring Consensus Diversity Since our MCC can generate multiple consensuses, we also measure the difference diversity/different consensuses. Given a set of n consensuses $\{P_1, \cdots, P_n\}$, let $ARI(P_i, P_j)$ and $NMI((P_i, P_j)$ denote the adjusted Rand index [30] and normalized mutual information [31] between two consensuses P_i and P_j. We use the following two measures to compute their diversity:

$$D1 = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (1 - ARI(P_i, P_j)), \tag{6.8}$$

$$D2 = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (1 - NMI(P_i, P_j)). \tag{6.9}$$

Note that $D1$ and $D2$ measure the pair-wise consensus similarity using adjusted rand Index and normalized mutual information respectively. The larger the measures, the more diverse the consensuses are.

6.4 Results Analysis We compare the performance of MMC with other single consensus clustering algorithms including K-means on the consensus similarity matrix (KC) [25], the NMF-based consensus clustering [3], the cluster-based similarity partitioning algorithm (CSPA) [24], the HyperGraph Partitioning Algorithm (HGPA) [24], and the weighted consensus clustering algorithm [25]. Since MCC generates multiple consensuses, we use HighMCC, AveMCC and LowMCC to denote the highest accuracy, the average accuracy, and the lowest accuracy of the multiple consensuses generated by MCC, respectively. The experiment comparisons are shown in Table 2. We also include the results of K-means on the original datasets as the baselines. From the comparison, we observe that 1) consensus clustering algorithms generally improve K-means clustering on all the datasets expect Reuters, and 2) HighMCC has the best performance on all the datasets. And it verifies that a consensus obtained from a group of clusterings might have better performance than the consensus obtained from all the input clusterings. Table 3 presents the diversity measurement of the multiple consensuses generated by our MCC framework.

Table 3: Diversity measurement.

Dataset	D1	D2
Glass	0.547	0.485
Ionosphere	0.501	0.416
Iris	0.442	0.408
Soybean	0.655	0.498
Wine	0.442	0.2238
Zoo	0.368	0.2851
CSTR	0.577	0.528
Log	0.664	0.538
LetterIJL	0.452	0.371
Reuters	0.451	0.322
WebKB4	0.577	0.445

Comparing Multiple Clustering Views: Then, we use the datasets of Glass and Wine as examples to show how effective our MCC approach in presenting multiple clustering views. Figure 4 displays the similarities of three clustering results (C1, C2, C3) generated by MCC and the ground truth (T). The similarities between different clusterings are computed using Equation 3.2. From Figure 4, it could be observed that each clustering has a relative small distance with the true labels, but with a higher distance among themselves. In other words, the three clustering views have good quality values and they are quite different from each other.

Explore Multiple Clustering Views: To further explore multiple clustering views, we also present the data sets in a 2-D mapping using Multi-Dimensional Scaling (MDS) [32] and Principal Components Analysis (PCA) [33]. Figure 5 and Figure 6 show the 2D-mapping plots of different clustering views on the Glass and Wine datasets, respectively. In these figures, each node is colored, and shaped based on the clustering indicators. We can clearly observe that the clustering views are quite different.

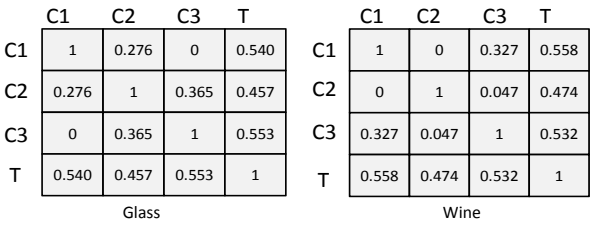

Figure 4: Similarity comparisons of three clustering views.

7 A Case Study of Protein-Protein Interaction

7.1 Introduction Proteins interact with other proteins either in pairs or as components of larger complexes in many cellular processes, including metabolism, endocrine and exocrine, signaling, synthesis, and transport. Identifying functional modules from PPI networks is an important and challenging task in post genomic era. In general, the PPI network can be represented as a graph, where the nodes represent the proteins and the edges indicate the interaction between two proteins. Cluster analysis is a popular methodology for the extraction of function modules from protein interaction networks since it has been observed by biologists that groups of highly interacting proteins could be involved in common biological processes [34][35][36]. In this section, we present a case study of exploring multiple clustering views of protein-protein interaction data using our MCC framework.

7.2 Dataset Description MIPS Yeast Protein-Protein Interaction Database, which is available at http://mips.gsf.de/proj/ppi, is used in our case study. The dataset is a collection of manually created high-quality PPI data collected from the scientific literature by expert curators. It consists of 8617 interactions between 871 proteins.

Table 2: The experiment comparisons on 11 data sets.

Datasets	Kmeans	KC	CSPA	HPGA	NMFC	WC	HighMCC	AvgMCC	LowMCC
Glass	38%	45%	43%	40%	49%	49%	50%	47%	40%
Ionosphere	70%	71%	68%	52%	71%	71%	71%	64%	52%
Iris	83%	72%	79.29%	86%	69%	89%	89%	84%	75%
Soybean	72%	82%	70%	81%	89%	91%	93%	87%	70%
Wine	68%	68%	69%	52%	70%	72%	72%	65%	55%
Zoo	61%	59%	56%	58%	62%	70%	71%	68%	64%
CSTR	45%	37%	50%	62%	56%	64%	64%	55%	47%
Log	61%	77%	47%	43%	71%	69%	81%	67%	65%
LetterIJL	49%	48%	48%	53%	52%	52%	53%	52%	51%
Reuters	45%	44%	43%	44%	43%	44%	45%	45%	45%
WebKB4	60%	56%	61%	62%	64%	63%	63%	60%	50%

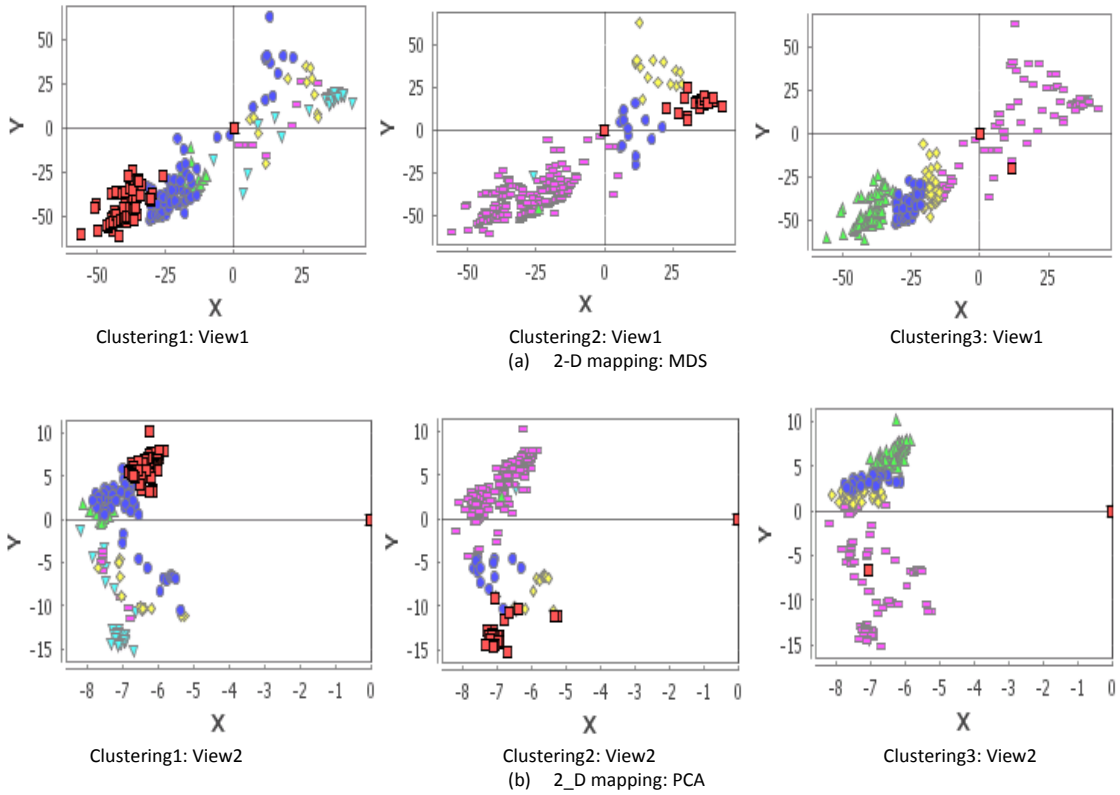

Figure 5: Multiple clustering views for the Glass dataset.

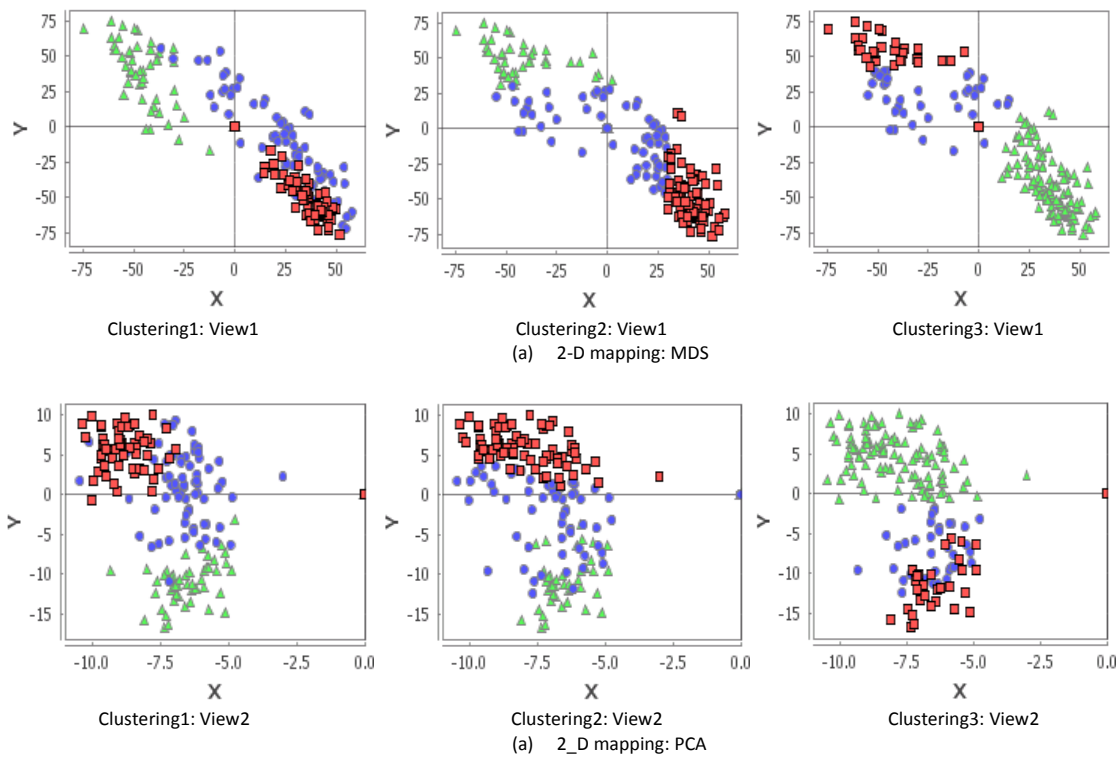

Figure 6: Multiple clustering views for the Wine dataset.

7.3 Base Clustering Generation We use four conventional graph clustering algorithms including three methods in Metis (rbr, direct, mlkp) [1] , and spectral clustering to obtain the base clusterings. The algorithms are described below [34]:

1. Repeated bisections (rbr): it is a top-down clustering algorithm, which uses a sequence of $k-1$ repeated bisections (k is number of clusters) to compute the desired k-way clustering solution;

2. Direct k-way partitioning (direct): in this algorithm, k instances are selected from the dataset as the seeds of the k clusters. Then the algorithm assign each instance to the cluster corresponding to its most similar seed by computing the similarity with these k seeds. This is a method that is computed by simultaneously finding k clusters;

3. Multilevel k-way partitioning (mlkp): it is a multilevel partitioning algorithm and works in three phases as coarsening, initial partitioning, and refinement;

4. Spectral clustering: a spectral clustering algorithm is obtained by recursively applying a spectral method for graph partitioning [37].

7.4 Evaluation Measures Modularity and normalized mutual information are used for performance evaluation in the case study.

Modularity: we apply modularity to evaluate our methods in the topology-based scenario. More linking edges between the nodes in the same clusters, and less linking edges between the nodes in the difference clusters would lead higher modularity value.

Normalized Mutual Information Second, we using NMI to compare the performance of each algorithm, which was originally described in [23] [34].

(7.10)
$$\phi^{NMI}(\lambda^a, \lambda^b) = \frac{2}{n} \times \sum_{l=1}^{k^a} \sum_{h=1}^{k^b} n_l^h \times \log_{k^a \times k^b} \frac{n_l^h \times n}{n^h \times n_l},$$

where λ^a,λ^b are the calculated labels for each instance, and the ground truth for them respectively. k_a is the number of clusters in λ^a, k_b is the number of clusters in λ^b. n_l is the number of instances in cluster l in λ^a, n_h is the number of instances in cluster h in λ^b, n is the number of instances in both cluster l in and cluster h in λ^b . n the number of all instances. From the equation

[1] The software can be downloaded from http://glaros.dtc.umn.edu/gkhome/views/metis/.

above, if λ^a, λ^b are exactly the same, $\phi^{NMI}(\lambda^a, \lambda^b)$ should be 1.

7.5 Results Analysis Figure 7 and Figure 8 show the performance comparison of MCC with other consensus clustering algorithms (WClustering, PCA, CSPA, HGPA) using modularity and NMI respectively. *Consensus-All* denotes the result which is obtained by combining all input clusterings using the selected consensus clustering algorithm.

From Figure 7 and Figure 8, we observe that no matter which consensus clustering algorithm is used, HighMCC (i.e., the best performance of multiple consensuses generated by our MCC) always achieves the best performance.

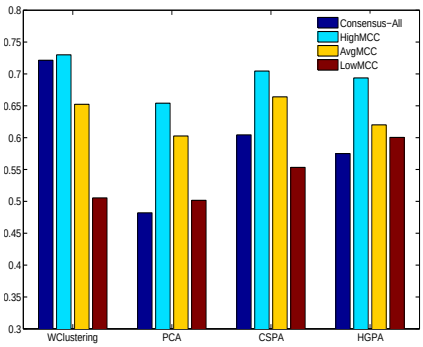

Figure 7: Modularity comparison on the PPI data set.

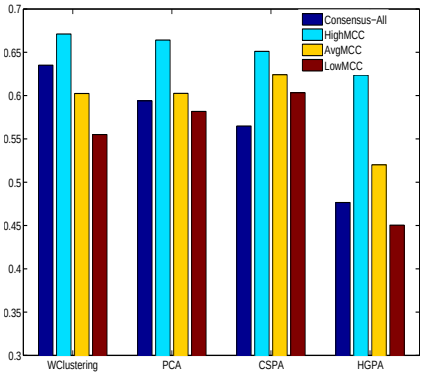

Figure 8: NMI comparison on the PPI data set.

7.6 Domain-based Evaluation [34] We also use known biological association from the Gene Ontology (GO) [42] Consortium Online Database to test whether the clusters obtained from our experiments correspond to known functional modules. The GO dataset provides the information including cellular component, molecular

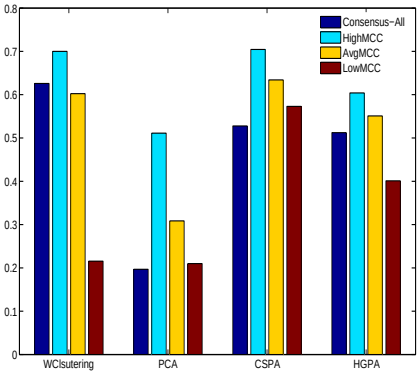

Figure 9: Clustering score comparison.

level and biological process, and we only use the information about biological process which refers to entities at both the cellular and organism levels of granularity to calculate the P-value and clustering score.

P-value comparison: P-value can be used to calculate the statistical and biological significance of a cluster of proteins. In our case study, we evaluate P-value for each annotation in each cluster for each algorithm. Firstly, we assume the cluster we evaluate is with the size n and it contains m proteins in the particular biological annotation noted in GO database. For each cluster in each algorithm we choose the smallest value of P-value as the cluster's P-value. P-value is computed as follows:

$$\text{P-value} = \sum_{i=m}^{n} \frac{\binom{M}{i}\binom{N-M}{n-i}}{\binom{N}{n}}, \tag{7.11}$$

where N is the total number of proteins in GO database; M is the number of proteins in GO database with particular biological annotation. Now, we set cutoff value (which will be explained in clustering score comparison) as 0.05 and get the significant clusters for each algorithm. For all algorithms, the minimum number of significant clusters is 31, so, in Figure 10, we only show the top 31 significant clusters for each algorithm. And from this figure, we observe that using the best clustering result from MCC, the performance could be significantly improved in the value of $-Plog(P)$.

Clustering-score evaluation: Then we use clustering-score to evaluate the clustering results for each algorithm [34]. A *cutoff* value is used to differentiate significant cluster from the insignificant cluster. If the P-value of a cluster is greater than *cutoff*, then this cluster is an insignificant cluster. The clustering score is defined as follows [34]:

$$CScore = 1 - \frac{\sum_{i=1}^{n_S} \min(p_i) + (n_I \times cutoff)}{(n_S + n_I) \times cutoff}, \tag{7.12}$$

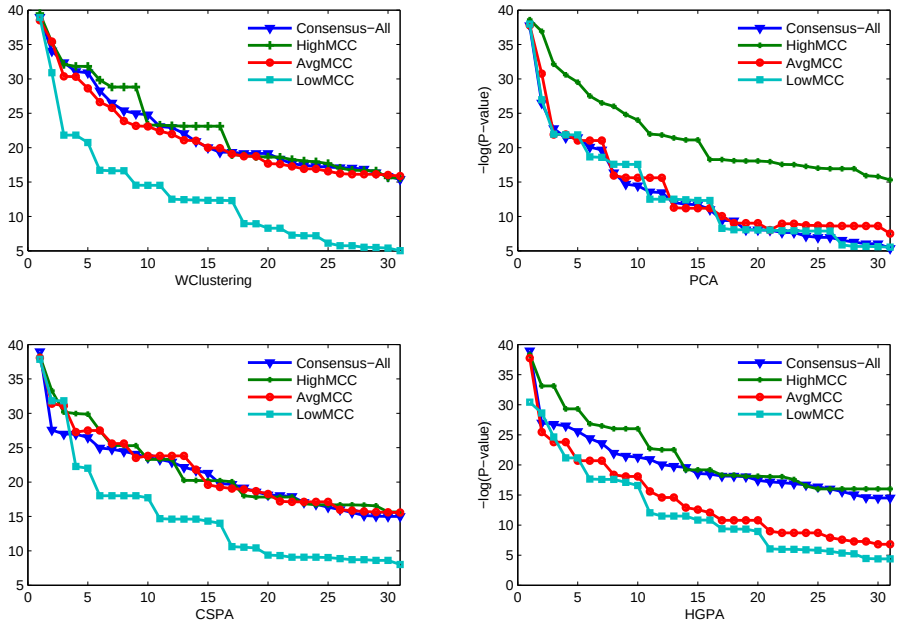

Figure 10: P-value comparison.

where n_S is the number of clusters which are significant, n_I is the number of clusters which are insignificant, and $\min(p_i)$ denotes the smallest P-value of the significant cluster i. Generally, a larger clustering-score means a better clustering result. Figure 9 also validates the effectiveness of MCC.

8 Comparison between MCC and Meta Clustering

In the section, we compare the performance of MCC with meta clustering. Since there is no clustering results generated from meta clustering, we extend it by performing consensus clustering on each resulting group. In particular, in our experiments, we first apply meta-clustering [8] to partition all input clustering results into three groups, and then use weighted consensus clustering to generate clustering results for each group. To verify the effectiveness of MCC, we compare the highest accuracy of MCC and meta clustering. Table 4 shows the accuracy on 11 datasets and Table 5 shows the performance comparison on the PPI dataset. From the tables, we observe that MCC outperforms simple extension of meta clustering.

Table 4: Accuracy comparison between MCC and Meta Clustering.

Datasets	HighMCC	HighMetaC
Glass	50%	45%
Ionosphere	71%	71%
Iris	89%	85%
Soybean	93%	85%
Wine	72%	69%
Zoo	71%	63%
CSTR	64%	56%
Log	81%	74%
LetterIJL	53%	51%
Reuters	45%	45%
WebKB4	63%	63%

9 Related Work

9.1 Consensus Clustering Given multiple partitions generated by different clustering algorithms or different subsets of the dataset or different feature spaces, consensus clustering aims to "combine" them into a single consolidated clustering that maximizes the agreement shared among all available clustering solutions and consequently obtains a better clustering solution. Many approaches have been developed to solve ensemble clustering problems over recently years [1][2][3][4][5]. Different from traditional consensus clustering, our MCC groups the input clusterings and obtains multiple consensuses (a consensus for each group).

9.2 Meta Clustering Recently, meta clustering is proposed to generate many alternative good clusterings of the data and allows the users to select the useful clusterings [8]. In particular, meta clustering groups similar input clusterings together so that users only need to ex-

Table 5: Performance comparison between MCC and Meta Clustering on PPI dataset.

Performance Measure	HighMCC	HighMetaC
Modularity	73%	67%
NMI	67%	60%
ClusteringScore	71%	61%

amine a small number of different clusterings. Different from meta clustering that finds many alternative good clusterings of the data, our MCC generates consensus clusterings from the input clusterings of a given data set.

9.3 Alternative Clustering Recently, many techniques have been proposed to find alternative clusterings or multiple complementary clusterings [9][38]. For example, Qi and Davidson [38] proposed a principled framework for finding alternative clusterings by transforming the input data so that new clustering can be found in the transformed space. Cui et al. presented a framework to find all non-redundant clusterings of the data where data points of one cluster can belong to different clusters in other views [9]. Different from alternative clustering, our MCC aims to explore multiple clusterings views from the input clusterings of a given data set.

9.4 Hierarchical Clustering Hierarchical Clustering algorithms are unsupervised methods to generate tree-like clustering solutions. They group the data points into a hierarchical tree structure using bottom-up or top-down approaches [39]. The typical bottom-up approaches take each data point as a single cluster to start off with and then build bigger clusters by grouping similar clusters together until the entire data set is encapsulated into one final cluster. The divisive approaches start with all data points in one cluster and then split the larger cluster recursively. Many research efforts have been reported on algorithm-level improvements of the hierarchical clustering process and on understanding of hierarchical clustering [40][41]. In our work, we use hierarchical clustering to generate a tree structure of the input clustering and to support the exploration of multiple clustering views.

9.5 Cluster Ensemble Selection The problem of selecting a subset of input clusterings to form a smaller but better performing cluster ensemble than using all available solutions has been studied recently for consensus clustering [7][6]. Different from ensemble selection, our MCC groups all the input clusterings into different groups and generates different consensuses for different groups.

10 Conclusion

In this paper we develop MCC to explore multiple clustering views of a given dataset from a set of input clusterings. Given a set of input clusterings of a particular data set, it first employs hierarchical clustering to cluster the input clusterings and then uses consensus clustering to generate a consensus for each cluster of the input clusterings. Extensive experimental results as well as a case study demonstrate the effectiveness of our proposed method.

11 Acknowledgment

The work is partially supported by NSF grants DBI-0850203 and HRD-0833093.

References

[1] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," in *ICDE*, 2005, pp. 341–352.

[2] A. Strehl, J. Ghosh, and C. Cardie, "Cluster ensembles - a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2002.

[3] T. Li, C. Ding, and M. I. Jordan, "Solving consensus and semi-supyervised clustering problems using nonnegative matrix factorization," in *ICDM*, 2007, pp. 577–582.

[4] X. Hu, I. Yoo, X. Zhang, P. Nanavati, and D. Das, "Wavelet transformation and cluster ensemble for gene expression analysis," *International Journal of Bioinformatics Research and Applications*, vol. 1, no. 4, pp. 447 – 460, 2005.

[5] X. Z. Fern, C. E. Brodley, X. Z. Fern, and C. E. Brodley, "Solving cluster ensemble problems by bipartite graph partitioning," in *ICML*, 2004.

[6] X. Z. Fern and W. Lin, "Cluster ensemble selection," *Journal of Statistical Analysis and Data Mining*, vol. 1, no. 3, pp. 128 – 141, 2008.

[7] J. Azimi and X. Fern, "Adaptive cluster ensemble selection," in *In Proceedings of International Joint Conference on Artificial Intellegence*, 2009, pp. 993–997.

[8] R. Caruana, M. Elhawary, and N. Nguyen, "Meta clustering," in *ICDM*, 2006.

[9] Y. Cui, X. Z. Fern, and J. Dy, "Non-redundant multi-view clustering via orthogonalization," in *ICDM*, 2007, pp. 133–142.

[10] C. Ding and X. He, "Cluster merging and splitting in hierarchical clustering algorithms," in *ICDM*, 2002, pp. 139–146.

[11] B.-H. A, E. A, and G. I, "A stability based method for discovering structure in clustered data," *Pacific Symposium on Biocomputing*, 2002.

[12] S. V. Dongen and S. V. Dongen, "Performance criteria for graph clustering and markov cluster experiments," National Research Institute for Mathematics and Computer Science, Tech. Rep. INS-R0012, 2000.

[13] M. Meila, "Comparing clusterings," Statistics, University of Washington, Tech. Rep., 2002.

[14] C. L. Mallows, "A note on asymptotic joint normality," *The Annals of Mathematical Statistics*, vol. 43, no. 2, pp. 508–515, 1972.

[15] D. Zhou, J. Li, and H. Zha, "A new mallows distance based metric for comparing clusterings," *ICML*, pp. 1028–1035, 2005.

[16] S. Navlakha, R. Rastogi, and N. Shrivastava, "Graph summarization with bounded error," in *In Proceedings of the 2008 ACM SIGMOD Conference*, 2008, pp. 419–432.

[17] M. E. J. Newman, "Modularity and community structure in networks," in *In Proceedings of the National Academy of Sciences*, 2006, pp. 8577–8582.

[18] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," *IEEE Transactions on In Knowledge and Data Engineering*, vol. 20, no. 2, December 2007.

[19] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Phys Rev E Stat Nonlin Soft Matter Phys*, vol. 74, no. 1, 2006.

[20] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, Feb 2004.

[21] S. Navlakha, J. White, N. Nagarajan, M. Pop, and C. Kingsford, "Finding biologically accurate clusterings in hierarchical tree decompositions using the variation of information," in *Proceedings of the 13th Annual International Conference on Research in Computational Molecular Biology*, 2009, pp. 400 – 417.

[22] S. Asur, D. Ucar, and S. Parthasarathy, "An ensemble framework for clustering protein-protein interaction networks," *Bioinformatics*, vol. 23, no. 13, pp. i29–i40, 2007.

[23] A. Strehl and J. Ghosh, "Relationship-based clustering and visualization for high-dimensional data mining," *INFORMS Journal on Computing*, vol. 15, no. 2, pp. 208–230, 2003.

[24] A. Strehl, J. Ghosh, and C. Cardie, "Cluster ensembles - a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2002.

[25] T. Li and C. Ding, "Weighted consensus clustering," in *In Proceedings of 2008 SIAM International Conference on Data Mining*, 2008, pp. 798–809.

[26] C. L. Blake and C. J. Merz, 1998, UCI Repository of Machine Learning Databases.

[27] A. K. McCallum, "Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering," 1996, http://www.cs.cmu.edu/ mccallum/bow.

[28] Multigrid multidimensional scaling E.-H. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore, "Webace: a web agent for document categorization and exploration," in *In Proceedings of the second international conference on Autonomous agents*, 1998, pp. 408–415.

[29] T. Li and C. Ding, "The relationships among various nonnegative matrix factorization methods for clustering," in *ICDM*, pp. 362 – 371, 2006.

[30] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971.

[31] A. L. Fred and A. K. Jain, "Robust data clustering," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, p. 128, 2003.

[32] M. M. Bronstein, A. M. Bronstein, R. Kimmel, and I. Yavneh, "Multigrid multidimensional scaling," in *Numerical Linear Algebra with Applications (NLAA), March-April*, 2006, vol. 13, pp. 149–171.

[33] J. Shlens, "A tutorial on principal component analysis," Center for Neural Science, New York University, Tech. Rep., 2009.

[34] S. Asur, S. Parthasarathy, and D. Ucar, "An ensemble framework for clustering protein-protein interaction networks," *Bioinformatics*, vol. 23, no. 13, 2007.

[35] Y. Zhang, E. Zeng, T. Li, and G. Narasimhan, "Weighted consensus clustering for identifying functional modules in protein-protein interaction networks," in *The 8th International Conference on Machine Learning and Applications*, 2009, pp. 539–544.

[36] S. Zhang, X. Ning, and X.-S. Zhang, "Identification of functional modules in a ppi network by clique percolation clustering," *Journal of Computational Biology and Chemistry*, vol. 30, no. 6, pp. 445–451, 2006.

[37] U. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395-416,2007.

[38] Z. Qi and I. Davidson, "a principled and flexible framework for finding alternative clusterings," in *ACM SIGKDD Conference On Knowledge Discovery and Data Mining*, 2009, pp. 717–726.

[39] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Addison-Wesley Longman Publishing, 2005.

[40] J. Wu, H. Xiong, and J. Chen, "Towards understanding hierarchical clustering: A data distribution perspective," *Neurocomputing*, vol. 72, no. 10-12, pp. 2319–2330, 2009.

[41] Y. Zhao and G. Karypis, "Evaluation of hierarchical clustering algorithms for document datasets," in *Conference of Information and Knowledge Management*, 2002, pp. 515–524.

[42] M. Ashburner and C. Ball and J. Blake and D. Botstein and H. Butler and J. Michael and A. Davis and K. Dolinski andS. Dwight and J. Eppig and M. Harri and D. Hill and L. Traver and A. Kassarskis and S. Levis and J. Matese and E. Richardson and M. Ringwald and G. Rubin and G. Sherlock, "Gene ontology: tool for the unification of biology," in *Nat. Genet*, 2000, vol. 25, pp. 24-29.