Solving Cluster Ensemble Problems by Bipartite Graph Partitioning

Xiaoli Zhang Fern XZ@ECN.PURDUE.EDU
Carla E. Brodley BRODLEY@ECN.PURDUE.EDU
School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907

Abstract

A critical problem in cluster ensemble research is how to combine multiple clusterings to yield a final superior clustering result. Leveraging advanced graph partitioning techniques, we solve this problem by reducing it to a graph partitioning problem. We introduce a new reduction method that constructs a bipartite graph from a given cluster ensemble. The resulting graph models both instances and clusters of the ensemble simultaneously as vertices in the graph. Our approach retains all of the information provided by a given ensemble, allowing the similarity among instances and the similarity among clusters to be considered collectively in forming the final clustering. Further, the resulting graph partitioning problem can be solved efficiently. We empirically evaluate the proposed approach against two commonly used graph formulations and show that it is more robust and achieves comparable or better performance in comparison to its competitors.

1. Introduction

Clustering for unsupervised data exploration and analysis has been investigated for decades in the statistics, data mining, and machine learning communities. A recent advance of clustering techniques is the development of cluster ensemble or consensus clustering techniques (Strehl & Ghosh, 2002; Fern & Brodley, 2003; Monti et al., 2003; Topchy et al., 2003), which seek to improve clustering performance by first generating multiple partitions of a given data set and then combining them to form a final (presumably superior) clustering solution. Such techniques have been shown to provide a generic tool for improving the performance of basic clustering algorithms.

A critical problem in designing a cluster ensemble system is how to combine a given ensemble of clusterings in order to produce a final solution, referred to as the *cluster ensemble problem* here. In this paper we approach this problem by reducing it to a graph partitioning problem. In graph partitioning, the input is a graph that consists of vertices and weighted edges. The goal is to partition the graph into K roughly equal-sized parts with the objective of minimizing the cut (the sum of the weights of those edges connecting different parts). We choose to solve cluster ensemble problems using graph partitioning techniques for two reasons. First, graph partitioning is a well studied area and algorithms such as spectral clustering have been successful in a variety of applications (Shi & Malik, 2000; Dhillon, 2001). Second, cluster ensembles provide a natural way to define similarity measures for computing the weight of the edges in a graph, which is an important and sometimes hard to satisfy prerequisite for the success of graph partitioning techniques (Bach & Jordan, 2004).

Previously, Strehl and Ghosh (2002) proposed two approaches to formulating graph partitioning problems for cluster ensembles. The first formulation is an instance-based approach that models instances as vertices in a graph and computes the weight of each edge as the similarity between the pair of instances it connects based on how frequently they are clustered together. The second formulation is a cluster-based approach that models clusters as vertices and computes the weight of each edge as the similarity between the clusters based on the percentage of instances they share. Note that we cannot reconstruct the original cluster ensemble based on a graph formed by either the instance-based or cluster-based approach, indicating that both approaches incur information loss from a given ensemble. This paper proposes a new graph formulation that simultaneously models both instances and clusters as vertices in a bipartite graph. Such a graph retains all of the information of an ensemble, allowing both the similarity among instances and the similarity among clusters to be considered collectively to construct the final clusters. Moreover, the resulting

Appearing in *Proceedings of the 21^{st} International Conference on Machine Learning*, Banff, Canada, 2004. Copyright 2004 by the authors.

graph partitioning problem can be solved efficiently.

We experimentally compare the proposed graph formulation to the instance-based and cluster-based approaches on five data sets. To understand the impact of different cluster ensemble types, two different approaches of generating cluster ensembles are applied. Our experiments show that among the three formulations, the proposed bipartite approach achieves the most robust clustering performance. For each data set, the bipartite formulation is always comparable and in some cases significantly better than the other two approaches. In addition, we point out a natural connection between cluster ensemble problems and keyword-based document clustering. This connection raises interesting questions for cluster ensemble research and suggests directions for future research.

The remainder of this paper is arranged as follows. Section 2 introduces the basics of cluster ensembles followed by a brief review of the related work. Section 3 introduces the problem of graph partitioning. In Section 4 we describe the instance-based and cluster-based graph formulations. Our bipartite graph formulation is then presented in Section 5. In Section 6 we describe our experimental design and present the results in Section 7. Finally, Section 8 concludes the paper with an overview of the contributions and a discussion of the connection between cluster ensemble problems and document clustering.

2. Cluster Ensembles and Related Work

This section introduces the basics of cluster ensembles and briefly reviews the existing techniques for solving cluster ensemble problems that do not involve graph partitioning.

2.1. Cluster Ensembles

A cluster ensemble system solves a clustering problem in two steps. The first step takes a data set as input and outputs an ensemble of clustering solutions. The second step takes the cluster ensemble as input and combines the solutions to produce a single clustering as the final output. In this paper we assume hard clusterings are used to form the ensembles. But it should be noted that all of the graph-formulation approaches examined in this paper can be applied to cluster ensembles with soft clusterings, directly or with minor modifications. Below we formally define cluster ensembles and state the cluster ensemble problem.

Given a data set $X = \{X_1, X_2, \cdots, X_n\}$, a *cluster ensemble* is a set of clustering solutions, represented as $C = \{C^1, C^2, \cdots, C^R\}$, where R is the ensemble size, i.e., the number of clusterings in the ensemble. Each clustering solution C^r is simply a partition of the data set X into K_r disjoint clusters of instances, represented as $C^r = \{C_1^r, C_2^r, \cdots, C_{K_r}^r\}$, where $\cup_k C_k^r = X$.

Generally speaking, the value of K_r for different clustering runs can be either the same or different. The techniques we study here can be applied in both cases.

Given a cluster ensemble C and a number K, the desired number of clusters, to solve the cluster ensemble problem is to use the information provided by C and partition X into K disjoint clusters as the final clustering solution. Note that in some cases, the original features of X may also be used with C to produce the final clustering solution. This study focuses on the case where X is only used to generate the ensemble.

2.2. Related Work on Combining Clusterings

While our paper focuses on combining clusterings by graph partitioning, other alternative approaches exist. A commonly used approach (Fred & Jain, 2002; Fern & Brodley, 2003; Monti et al., 2003) combines the clusterings by first generating a similarity matrix for instances and then applying agglomerative clustering algorithms to produce a final clustering. Recently Topchy et al. (2003; 2004) propose to represent a cluster ensemble as a new set of features describing the instances and produce final clusters by applying K-means and EM to the new features. See Dimitriadou et al., 2001 and Dudoit & Fridlyand, 2003 for other representative technqiues for combining clusterings.

While performing a thorough comparison of all available techniques is beyond the scope of this paper, in recent studies (Strehl & Ghosh, 2002; Topchy et al., 2004), graph-partitioning based approaches appear to be highly competitive compared to other techniques.

3. Graph Partitioning

This section describes the basics of graph partitioning. A weighted graph is represented by $G = (V, W)$, where V is a set of vertices and W is a nonnegative and symmetric $|V| \times |V|$ similarity matrix characterizing the similarity between each pair of vertices.

The input to a graph partitioning problem is a weighted graph G and a number K. To partition a graph into K parts is to find K disjoint clusters of vertices $P = \{P_1, P_2, \cdots, P_K\}$, where $\cup_k P_k = V$. Unless a given graph has K, or more than K, strongly connected components, any K-way partition will cross some of the graph edges. The sum of the weights of these crossed edges is defined as the cut of a partition

P: $Cut(P, W) = \sum W(i, j)$, where vertices i and j do not belong to the same cluster.

The general goal of graph partitioning is to find a K-way partition that minimizes the cut, subject to the constraint that each part should contain roughly the same number of vertices.[1] In practice, various graph partitioning algorithms define different optimization criteria based on the above goal. Examples include the normalized cut criterion (Shi & Malik, 2000) and the ratio cut criterion (Hagen & Kahng, 1992). See (Fjallstrom, 1998) for an in-depth discussion. Here we defer the discussion of our choice of graph partitioning algorithm to Section 6.2. Given the basics of cluster ensembles and graph partitioning, we are now ready to explore various techniques for reducing a cluster ensemble problem to a graph partitioning problem.

4. Existing Graph Formulations for Cluster Ensembles

This section introduces two existing techniques proposed by Strehl and Ghosh (2002) for formulating graphs from cluster ensembles. We rename these two techniques as instance-based and cluster-based approaches to characterize the differences between them.

4.1. Instance-Based Graph Formulation

Instance-Based Graph Formulation (IBGF) constructs a graph to model the pairwise relationships among instances of the data set X. Recall that the commonly used agglomerative approach (Fred & Jain, 2002; Fern & Brodley, 2003; Monti et al., 2003) generates a similarity matrix from the cluster ensemble and then performs agglomerative clustering using the similarity matrix. IBGF uses this matrix in conjunction with graph partitioning. Below we formally describe IBGF.

Given a cluster ensemble $C = \{C^1, \cdots, C^R\}$, IBGF constructs a fully connected graph $G = (V, W)$, where

- V is a set of n vertices, each representing an instance of X.
- W is a similarity matrix and $W(i, j) = \frac{1}{R}\sum_{r=1}^{R} I(g_r(X_i) = g_r(X_j))$, where $I(\cdot)$ is an indicator function that returns 1 if the argument is true and 0 otherwise; $g_r(\cdot)$ takes an instance and returns the cluster that it belongs to in C^r.

$W(i, j)$ measures how frequently the instances i and j are clustered together in the given ensemble. In recent work (Fern & Brodley, 2003; Monti et al., 2003), this similarity measure has been shown to give satisfactory performance in domains where a good similarity (or distance) metric is otherwise hard to find. Once a graph is constructed, one can solve the graph partitioning problem using any graph partitioning technique and the resulting partition can be directly output as the final clustering solution.

Note that IBGF constructs a fully connected graph, resulting in a graph partitioning problem of size n^2, where n is the number of instances. Depending on the algorithm used to partition the graph, the computational complexity of IBGF may vary. But generally it is computationally more expensive than the cluster-based approach and our proposed approach, which is a key disadvantage of IBGF.

4.2. Cluster-Based Graph Formulation

Note that clusters formed in different clusterings may contain the same set of instances or largely overlap with each other. Such clusters are considered to be corresponding (similar) to one another. Cluster-Based Graph Formulation (CBGF) constructs a graph to model the correspondence (similarity) relationship among different clusters in a given ensemble and partitions the graph into groups so that the clusters of the same group correspond to one another.

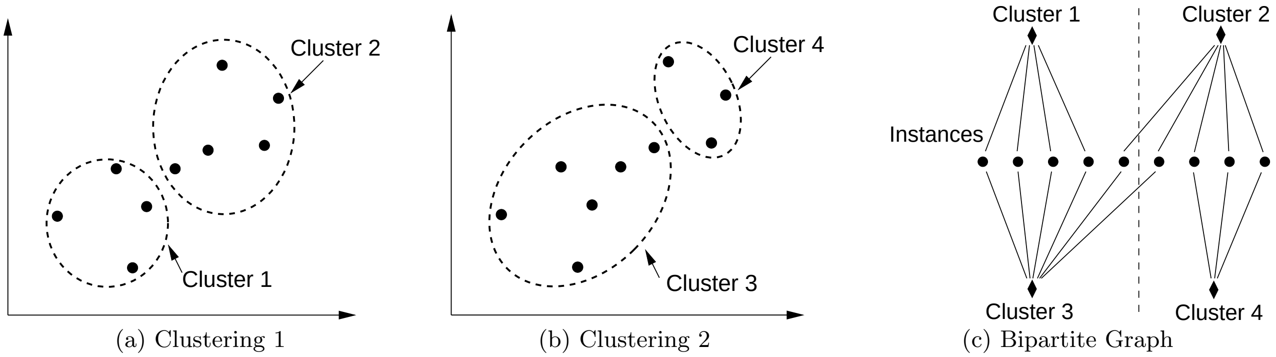
Given a cluster ensemble $C = \{C^1, \cdots, C^R\}$, we first rewrite C as $C = \{C_1^1, \cdots, C_{K_1}^1, \cdots, C_1^R, \cdots, C_{K_R}^R\}$ where C_j^i represents the jth cluster formed in the ith clustering run in the ensemble C. Denote the total number of clusters in C as $t = \sum_{r=1}^{R} K_r$. CBGF constructs a graph $G = (V, W)$, where

- V is a set of t vertices, each representing a cluster
- W is a matrix such that $W(i, j)$ is the similarity between the clusters C_i and C_j and is computed using the Jaccard measure as: $W(i, j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}$

Once a partition of the clusters is obtained, we can produce a final clustering of instances as follows. First we consider each group of clusters as a metacluster. For each clustering, an instance is considered to be *associated with* a metacluster if it contains the cluster to which the instance belongs. Note that an instance may be associated with different metaclusters in different runs, we assign an instance to the metacluster with which it is most frequently associated. Ties are broken randomly.

The basic assumption of CBGF is the existence of a correspondence structure among different clusters formed in the ensemble. This poses a potential problem — in cases where no such correspondence structure exists, this approach may fail to provide satisfactory performance. The advantage of CBGF is that it

[1] Note that in some cases this bias maybe unwarranted.



(a) Clustering 1 (b) Clustering 2 (c) Bipartite Graph

Figure 1. An example of the Hybrid Bipartite Graph Formulation

is computationally efficient. The size of the resulting graph partitioning problem is t^2, where t is the total number of clusters in the ensemble. This is significantly smaller than the n^2 of IBGF, assuming $t \ll n$.

Strehl and Ghosh (2002) also proposed a hypergraph-based approach, which models clusters as hyperedges and instances as vertices in a hypergraph and uses a hypergraph partitioning algorithm to produce a final partition. Conceptually, this approach forms a different type of graph and has the limitation that it can not model soft clustering. Practically, we observed that it performed worse than IBGF and CBGF on our data sets. Due to the above reasons, we choose not to further discuss this approach in this paper.

5. Hybrid Bipartite Graph Formulation

This section presents our Hybrid Bipartite Graph Formulation (HBGF) and explains its conceptual advantages over IBGF and CBGF.

Description of HBGF Given a cluster ensemble $C = \{C^1, \cdots, C^R\}$, HBGF constructs a graph $G = (V, W)$, where

- $V = V^C \cup V^I$, where V^C contains t vertices each representing a cluster of the ensemble; V^I contains n vertices each representing an instance of the data set X.
- W is defined as follows. If the vertices i and j are both clusters or both instances, $W(i, j) = 0$; otherwise if instance i belongs to cluster j $W(i, j) = W(j, i) = 1$ and 0 otherwise.

Note that W can be written as: $W = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$ where A is a connectivity matrix whose rows correspond to the instances and columns correspond to the clusters. $A(i, j)$ is an indicator that takes value 1 if instance i belongs to the j-th cluster and 0 otherwise.

Figure 1 shows an example of HBGF. Particularly, Figures 1(a) and (b) depict two different clusterings of nine instances and Figure 1(c) shows the graph constructed by HBGF, in which the diamond vertices represent the clusters and the round vertices represent the instances. An edge between an instance vertex and a cluster vertex indicates that the cluster contains the instance. All the edges in the graph have weight one (edges with zero weights are omitted from the graph). In this graph, cluster vertices are only connected to instance vertices and vice versa, forming a bipartite graph. If a new clustering is added to the ensemble, a new set of cluster vertices will be added to the graph and each of them will be connected to the instances that it contains.

Shown in Figure 1(c) as a dashed line, a partition of the bipartite graph partitions the cluster vertices and the instance vertices simultaneously. The partition of the instances can then be output as the final clustering.

Although HBGF's vertex set is the sum of IBGF's and CBGF's vertex sets, it can be shown (Dhillon, 2001) that due to its special structure, the real size of a bipartite graph partitioning problem is $n \times t$, where n is the number of instances and t is the total number of clusters in the ensemble C. This is significantly smaller compared to the size n^2 of IBGF, assuming that $t \ll n$.

Conceptual advantages of HBGF Comparing HBGF with IBGF and CBGF, we argue that it has two important conceptual advantages. First, the reduction of HBGF is lossless — the original cluster ensemble can be easily reconstructed from an HBGF graph. In contrast, IBGF and CBGF do not have this property.

To understand the second advantage of HBGF, it should be noted that IBGF and CBGF consider the similarity of instances and the similarity of clusters independently and, as shown below, such independent treatment may be problematic.

Comparing two pairs of instances (A, B) and (C, D), we assume that A and B are never clustered together in the ensemble and the same is true for pair (C, D). However, the instances A and B are each frequently clustered together with the same group of instances in the ensemble, i.e., A and B are frequently assigned to clusters that are similar to each other. In contrast, this is not true for C and D. Intuitively we consider A and B to be more similar to one another than C and D. However, IBGF will fail to differentiate these two cases and assign both similarities to be zero. This is because IBGF ignores the information about the similarity of clusters while computing the similarity of instances.

A similar problem exists for CBGF. For example, consider two pairs of clusters (C_1, C_2) and (C_3, C_4). Assume that $C_1 \cap C_2 = \phi$ and $C_3 \cap C_4 = \phi$. And further assume that the instances of C_1 and those of C_2 are often clustered together in other clustering runs, whereas this is not the case for C_3 and C_4. Note that CBGF will assign both similarities to zero while intuitively we would consider C_1 and C_2 to be more similar to one another than C_3 and C_4. CBGF fails to differentiate these two cases, because it does not take the similarity of instances into account.

In constrast, HBGF allows the similarity of instances and the similarity of clusters to be considered simultaneously in producing the final clustering. Thus it avoids the above problems of IBGF and CBGF.

6. Experimental Design

In this section we first describe the methods used in our experiments for generating cluster ensembles and then introduce the graph partitioning algorithms and how they are used in our experiments.

6.1. Generating Cluster Ensembles

Cluster ensembles can be generated in different ways. The resulting ensembles may differ and the same approach for solving the ensemble problems may perform differently accordingly. It is thus important for our experiments to consider different ways to generate cluster ensembles. Our experiments use two approaches, random subsampling (Dudoit & Fridlyand, 2003) and random projection (Fern & Brodley, 2003), to generate the ensembles. Note that for both approaches, K-means is used as the base clustering algorithm and the number K is pre-specified for each data set and remains the same for all clustering runs.

Note that we also examined a third approach, randomly restarting K-means, and it produced similar results to those of random subsampling. So we omit these results in the discussion of our experiments.

6.1.1. Random Subsampling

For each clustering run, we randomly subsample the original data set with a sampling rate of 70%. The subsampling is performed without replacement to avoid duplicating instances. We then cluster the subsample and assign each instance absent from the current subsample to its closest cluster based on its Euclidean distance to the cluster centers to ensure that all the instances are clustered in each clustering run.

6.1.2. Random Projection

For each clustering run, we first randomly generate a projection matrix $P_{d \times d'}$ to project the given data set onto a lower dimensional space, where d is the original dimension of the data set and d' is the dimension that we project the data onto. We then cluster the projected low-dimensional data set.

Recently, Fern and Brodley (2003) showed that both the diversity and quality of a cluster ensemble significantly impact what can be achieved by combining the clusterings of the ensemble. We choose the above two approaches because we expect them to produce ensembles with different properties. On one hand, we expect the clusterings generated by random projection to be diverse because it provides the base learner with different views of the data. On the other hand, we expect the quality of the clusterings produced by random subsampling to be higher because it provides the base learner with more complete information of the data.

6.2. Graph Partitioning Algorithms

Our goal is to evaluate different graph formulation approaches. To reduce the influence of any chosen graph partitioning algorithm on our evaluation, we use two well-known graph partitioning algorithms that differ with respect to their search for the best partition.

6.2.1. Spectral Graph Partitioning

Spectral graph partitioning is a well studied area with many successful applications. We choose a popular multi-way spectral graph partitioning algorithm proposed by Ng et al. (2002), which seeks to optimize the normalized cut criterion (Shi & Malik, 2000). We refer to this algorithm as SPEC.

SPEC can be simply described as follows. Given a graph $G = (V, W)$, it first computes the degree matrix D, which is a diagonal matrix such that $D(i, i) = \sum_j W(i, j)$. Based on D, it then computes a normalized weight matrix $L = D^{-1}W$ and finds L's K largest eigenvectors $u_1, u_2, \cdots, u_K$ to form matrix $U = [u_1, \cdots, u_K]$. The rows of U are then normalized to have unit length. Treating the rows of U as K-dimensional embeddings of the vertices of the graph, SPEC produces the final clustering solution by clustering the embedded points using K-means.

Intuitively, SPEC first embeds the vertices of a graph onto a K-dimensional space and then performs clustering in the K-dimensional space. For graphs generated by IBGF and CBGF, the clusters and instances are embedded and clustered separately. Interestingly, for HBGF, the clusters and instances are simultaneously embedded onto the same space and clustered together. Here we argue that this offers potential advantages over IBGF and CBGF. Compared to IBGF, the inclusion of the cluster vertices may help define the structure of the data and make it easier for K-means to find the structure in the K-dimensional space. In comparison to CBGF, it is expected to be more robust because even when the cluster vertices are not well structured, possibly due to the lack of a correspondence structure in the clusters, K-means can still perform reasonably well using the instance vertices.

6.2.2. Multilevel Graph Partition: Metis

Metis (Karypis & Kumar, 1998), a multilevel graph partitioning system, approaches the graph partitioning problem from a different angle. It partitions a graph using three basic steps: (1) coarsen the graph by collapsing vertices and edges; (2) partition the coarsened graph and (3) refine the partitions. In comparison to other graph partitioning algorithms, Metis is highly efficient and achieves competitive performance.

Comparing Metis and SPEC in practice, we observe mixed results — neither approach is consistently better than the other, indicating that both approaches have different advantages and weaknesses. Note that the goal of our experiments is to evaluate the different graph representations generated by IBGF, CBGF and HBGF, not the search bias of SPEC and Metis. To this end, we run both SPEC and Metis for each graph and report the maximum NMI (Our evaluation metric, See Section 7.2) obtained. This is done for all three types of graphs to ensure a fair comparison.

7. Experimental Results

The goal of the experiments is to evaluate the three graph formulations - IBGF, CBGF and HBGF - given different cluster ensembles.

Table 1. Summary of the data sets

DATA SET	EOS	GLASS	HRCT	ISOLET6	MODIS
#INST.	2398	214	1545	1440	4975
#CLASS	8	6	8	6	10
ORG. DIM.	20	9	183	617	112
RP DIM.	5	5	10	10	6
PCA DIM.	—	—	30	60	6

7.1. Data Sets and Parameter Settings

Five data sets are used in our experiments. The characteristics of the data sets are summarized in Table 1 with related parameter choices. HRCT is a high resolution computed tomography lung image data set with eight classes (Dy et al., 1999). MODIS and EOS are land cover data sets described by different feature sets. ISOLET6 and GLASS are from the UCI machine learning repository (Blake & Merz, 1998), where ISOLET6 is a subset of the ISOLET spoken letter recognition training set. In particular, ISOLET6 contains the instances of six classes (letters) randomly selected out of twenty six classes (letters).

To construct cluster ensembles using random projection, we need to specify the number of dimensions that random projection uses for each data set. The numbers are listed in the fifth row of Table 1.[2]

When random subsampling is used, for the EOS and GLASS data sets, we construct the cluster ensembles directly as described in Section 6.1.1. Note that the other three data sets are of high dimensionality. For these data sets, we reduce the dimension by Principal Component Analysis (PCA) prior to generating the ensembles. The dimensions that PCA uses are listed in the sixth row of Table 1. They are selected such that 85% of the data variance is preserved.

In all clustering runs, which include both the runs during ensemble construction and the partitioning of the final graphs, the cluster number K is set to be the same as the number of classes in a given data set. Note that the class labels are not used during clustering.

7.2. Evaluation Criterion

Because our data sets are labeled, we can assess the quality of the clustering solutions using external criteria that measure the discrepancy between the structure defined by a clustering and what is defined by the class labels. Here we choose to use an information theoretic criterion — the Normalized Mutual Information

[2] Random projection was first used for cluster ensembles by Fern and Brodley (2003), where EOS and HRCT were also used. We use the same settings for EOS and HRCT and arbitrarily selected the values for the other data sets.

(NMI) criterion (Strehl & Ghosh, 2002). Treating cluster labels and class labels as random variables, NMI measures the mutual information (Cover & Thomas, 1991) shared by the two random variables and normalizes it to a $[0, 1]$ range. Note that the expected NMI value of a random partition of the data is 0 and the optimal value 1 is attained when the class labels and cluster labels define the same partition of the data.

7.3. Results

Table 2 presents the performance of IBGF, CBGF and HBGF on cluster ensembles generated by both random subsampling (columns 2-4) and random projection (columns 5-7). For each ensemble generating method, we report the results with three different ensemble sizes (20, 40 and 60).[3] Each number reported here is obtained by averaging across ten random runs, where for each run we use both SPEC and Metis to partition the generated graph and record the maximum NMI obtained as the result of that run. Finally, the last row of each data set reports the average performance of the base learner for each ensemble type.

Comparing IBGF, CBGF and HBGF First let us look at the performance of the three approaches on cluster ensembles generated by random subsampling, shown in the first set of three columns of Table 2. We see that the performance of IBGF and CBGF are comparable for three of the five data sets (EOS, GLASS and MODIS) and mixed for the other two. In particular, IBGF performs better on the HRCT data set while CBGF is the winner for the ISOLET6 data set. When HBGF is used, we observe that in most cases its performance is comparable to the better of the other two approaches and is significantly better for EOS.

Similar observations can also be made for the random projection ensembles except that here CBGF appears to be less competitive and performs significantly worse than both IBGF and HBGF in two (EOS and HRCT) of the five data sets. We conjecture that this is because random projection ensembles tend to be more diverse[4] so that a correspondence structure is less likely to exist in the clusters.

Comparing with the base learner From Table 2, we observe that HBGF is the only graph formulation approach that leads to consistent performance improvement over the base learner in all cases. CBGF appears to be the least stable approach among the three graph formulations.

Table 2. Comparing IBGF, CBGF and HBGF

	RANDOM SUBSAMPL.			RANDOM PROJ.		
	20	40	60	20	40	60
			EOS			
IBGF	0.263	0.262	0.262	0.260	0.263	0.269
CBGF	0.262	0.264	0.263	0.246	0.247	0.247
HBGF	0.340	0.319	0.303	0.357	0.343	0.325
		(0.263)			(0.246)	
			GLASS			
IBGF	0.400	0.405	0.388	0.376	0.373	0.368
CBGF	0.393	0.398	0.395	0.379	0.378	0.377
HBGF	0.405	0.398	0.399	0.401	0.386	0.390
		(0.378)			(0.334)	
			HRCT			
IBGF	0.310	0.312	0.313	0.283	0.299	0.301
CBGF	0.279	0.277	0.280	0.256	0.267	0.274
HBGF	0.303	0.318	0.321	0.274	0.292	0.301
		(0.292)			(0.196)	
			ISOLET6			
IBGF	0.804	0.799	0.812	0.761	0.802	0.811
CBGF	0.832	0.837	0.833	0.750	0.790	0.802
HBGF	0.844	0.823	0.823	0.765	0.801	0.813
		(0.790)			(0.447)	
			MODIS			
IBGF	0.478	0.478	0.478	0.485	0.493	0.491
CBGF	0.476	0.478	0.478	0.482	0.490	0.491
HBGF	0.478	0.478	0.478	0.485	0.487	0.494
		(0.473)			(0.389)	

Interestingly, we see that both IBGF and CBGF fail to improve over the base learner for EOS when random subsampling ensembles are used, whereas HBGF achieves significantly better clustering results using the same cluster ensembles. We consider this as an indication that the constructed cluster ensembles indeed provide helpful information for clustering but was not captured by IBGF or CBGF in their graphs. We argue that HBGF's success may be attributed to the fact that it retains all the information of the ensembles and that it allows the similarity of instances and the similarity of clusters to be considered simultaneously.

Another interesting fact about the EOS data set is that for both types of cluster ensembles, the performance of HBGF degrades as the ensemble size increases. We plan to further explore this issue by inspecting the behavior of the graph partitioning algorithms.

From the above observations, we conclude that HBGF is more robust than both IBGF and CBGF. In our experiments, it yields competitive and sometimes better performance compared to both IBGF and CBGF.

[3] Other sizes were used but omitted to avoid redundancy.

[4] Our experiments confirm that random projection generates more diverse ensembles than random subsampling as measured by the approach of Fern and Brodley (2003).

8. Conclusions and Future Work

This paper presented HBGF, a graph formulation that reduces a cluster ensemble problem to a bipartite graph partitioning problem. HBGF simultaneously models the instances and clusters of a given ensemble as vertices of a bipartite graph. It achieves lossless reduction and allows the similarity among instances and the similarity among clusters to be considered simultaneously in thefinal clustering. Further, the resulting graph partitioning problem can be solved efficiently. We compare HBGF with two commonly used graph formulations for cluster ensembles. Our experiments on five data sets show that HBGF achieves comparable or better performance in comparison with the other two approaches and results in the most robust performance improvement over the base learner.

Finally, we want to point out an interesting connection between cluster ensemble problems and keyword-based document clustering. In document clustering, a set of keywords is used to represent a document as a vector. Each element of the vector indicates if the document contains a specific keyword. Considering an instance as a document and the clusters the instance belongs to as the keywords the document contains, clustering instances according to the clusters can be related to clustering documents based on the keywords. Naturally, we can relate CBGF to document clustering approaches based on clustering keywords (Slonim & Tishby, 2000) and HBGF to the bipartite co-clustering approach proposed by Dhillon (2001). This connection raises an interesting question. Can we borrow ideas from document clustering, a highly advanced area, to help solve cluster ensemble problems? This suggests two directions for future work. First, document clustering with keyword clustering has been shown to yield better performance than without keyword clustering (Slonim & Tishby, 2000). However, for cluster ensemble problems, CBGF appears to be inferior to both IBGF and HBGF. Such an inconsistency merits further exploration of the cluster-based approach. Secondly, our hybrid approach may also benefit from a recently proposed information theoretic co-clustering technique (Dhillon et al., 2003).

Acknowledgments

We thank the reviewers for their helpful comments. The authors were supported by NASA under Award number NCC2-1245.

References

Bach, F. R., & Jordan, M. I. (2004). Learning spectral clustering. *NIPS 16*.

Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. John Wiley & Sons.

Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. *KDD*.

Dhillon, I. S., Mallela, S., & Modha, D. S. (2003). Information-theoretic co-clustering. *KDD*.

Dimitriadou, E., Weingessel, A., & Hornik, K. (2001). Voting-merging: An ensemble method for clustering. *ICANN*.

Dudoit, S., & Fridlyand, J. (2003). Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, *19*.

Dy, J. G., Brodley, C. E., Kak, A., Shyu, C., & Broderick, L. S. (1999). The customized-queries approach to CBIR using EM. *CVPR*.

Fern, X. Z., & Brodley, C. E. (2003). Random projection for high dimensional data clustering: A cluster ensemble approach. *ICML*.

Fjallstrom, P. (1998). Algorithms for graph partitioning: A survey. Linkoping Electronic Articles in Computer and Information Science, 3.

Fred, A. L. N., & Jain, A. K. (2002). Data clustering using evidence accumulation. *ICPR*.

Hagen, L., & Kahng, A. (1992). New spectral methods for ratio cut partitioning and clustering. *IEEE transaction on CAD*, *11*, 1074–1085.

Karypis, G., & Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, *20*, 359–392.

Monti, S., Tamayo, P., Mesirov, J., & Golub, T. (2003). Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, *52*, 91–118.

Ng, A., Jordan, M., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. *NIPS 14*.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, *22*, 888–905.

Slonim, N., & Tishby, N. (2000). Document clustering using word clusters via the information bottleneck method. *Research and Development in Information Retrieval*.

Strehl, A., & Ghosh, J. (2002). Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Machine Learning Research*, *3*, 583–417.

Topchy, A., Jain, A. K., & Punch, W. (2003). Combining multiple weak clusterings. *ICDM*.

Topchy, A., Jain, A. K., & Punch, W. (2004). A mixture model for clustering ensembles. *Proc. of SIAM Conf. on Data Mining*.