# Consistent Bipartite Graph Co-Partitioning for Star-Structured High-Order Heterogeneous Data Co-Clustering

Bin Gao[1*], Tie-Yan Liu[2], Xin Zheng[3*], Qian-Sheng Cheng[1], and Wei-Ying Ma[2]

[1]LMAM, Dept. of Information Science, School of Mathematical Sciences, Peking University, Beijing, 100871, P. R. China
gaobin@math.pku.edu.cn
qcheng@pku.edu.cn

[2]Microsoft Research Asia
5F, Sigma Center, No. 49, Zhichun Road, Haidian District, Beijing, 100080, P. R. China
{t-tyliu, wyma}@microsoft.com

[3]Key Lab of Pervasive Computing, Dept. of Computer Science and Technology, Tsinghua University, Beijing 100084, P. R. China
zhengxin99@mails.tsinghua.edu.cn

## ABSTRACT

Heterogeneous data co-clustering has attracted more and more attention in recent years due to its high impact on various applications. While the co-clustering algorithms for two types of heterogeneous data (denoted by pair-wise co-clustering), such as documents and terms, have been well studied in the literature, the work on more types of heterogeneous data (denoted by high-order co-clustering) is still very limited. As an attempt in this direction, in this paper, we worked on a specific case of high-order co-clustering in which there is a central type of objects that connects the other types so as to form a star structure of the inter-relationships. Actually, this case could be a very good abstract for many real-world applications, such as the co-clustering of categories, documents and terms in text mining. In our philosophy, we treated such kind of problems as the fusion of multiple pair-wise co-clustering sub-problems with the constraint of the star structure. Accordingly, we proposed the concept of consistent bipartite graph co-partitioning, and developed an algorithm based on semi-definite programming (SDP) for efficient computation of the clustering results. Experiments on toy problems and real data both verified the effectiveness of our proposed method.

## Categories and Subject Descriptors

I.5.3 [**Pattern Recognition**]: Clustering – *algorithms.*

## General Terms

Algorithms, Performance, Design, Experimentation, Theory.

## Keywords

Co-clustering, High-Order Heterogeneous Data, Consistency, Spectral Graph.

## 1. INTRODUCTION

Clustering is a process that partitions a set of objects into groups or clusters such that objects in the same cluster are similar while objects in different clusters are dissimilar. Homogeneous data clustering has been a well-studied research area in the community of machine learning and data mining. Several algorithms have been developed including *k*-means [7], maximum likelihood estimation [7], spectral clustering [1][22] and so on.

In recent years, more and more data mining applications have asked for the clustering of highly inter-related heterogeneous objects, such as documents and terms in a text corpus, customers and purchasing items in market basket analysis and reviewers and movies in movie recommender systems. In such scenarios, using previous methods to cluster each type of objects independently might not work very well since the similarities among one type of objects sometimes can only be defined by the other type of objects. To tackle this problem, many researchers started to study the co-clustering of two types of heterogeneous data. Dhillon *et al* [4] and Zha *et al* [26] extended the traditional spectral clustering algorithms and proposed the bipartite spectral graph partitioning algorithm to co-cluster documents and terms simultaneously. Similar techniques were also applied in biology [14] and image processing [19]. Moreover, Dhillon *et al* [5] proposed the information theoretic co-clustering method based on mutual information.

As can be seen, the co-clustering of two types of heterogeneous objects (denoted by pair-wise co-clustering) has attracted much attention in recent years. However, comparatively speaking, the co-clustering of multiple types of objects (denoted by high-order co-clustering) has not been well studied in the literature. Some limited works include [23], [25] and so on. Take [23] for example, Zeng *et al* proposed a unified framework named ReCoM to cluster multi-type interrelated Web objects. However, there is no sounded objective function and theoretical proof on the effectiveness of this iterative algorithm.

For the above discussion, one may argue that the high-order co-clustering could be solved by trivially extending the pair-wise co-clustering methods. For example, one can use the spectral cut of a *k*-partite graph[1] to analyze the inter-relationship among the *k* types


*KDD'05*, August 21-24, 2005, Chicago, Illinois, USA.


[*] This work was performed when the first and the third authors were visiting students at Microsoft Research Asia.

[1] Here a *k*-partite graph is a graph whose graph vertices can be partitioned into *k* disjoint sets so that no two vertices within the same set are adjacent.

of objects. However, as will be seen in this paper later, such a trivial extension is only a pretty trap and we can not really get the desirable co-clustering results in such a way. Therefore, it is necessary to work on some more advanced technologies to handle the high-order co-clustering problems.

As a preliminary attempt to the high-order co-clustering problem, in this paper, we will work on a specific case of it in which there is a central type of objects that connects the other types so as to form a star structure of the inter-relationships (see Figure 1). Actually, this case could be a very good abstract of many real-world applications, such as Web users, search queries and Web pages in Web search systems (corresponding to Figure 1(a), where the query is the central data type), authors, conferences, papers, and key words in academic publications (corresponding to Figure 1(b), where paper is the central data type); customers, shops, shareholders, suppliers, and advertisement medias (corresponding to Figure 1(c), where shop is the central data type). Co-clustering over such heterogeneous data has its explicit meaning. For example, in an academic publication system, the co-clustering results might indicate that a certain group of authors usually write research papers of a certain series of topics using a certain list of key words, and submitted to a certain kind of conferences.

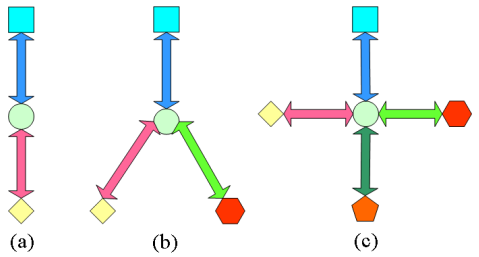


**Figure 1. The Star-structured High-order Heterogeneous Data.**

It is easy to understand that the basic element of the star structure is the triplet as shown in Figure 1(a). If we can successfully co-cluster such triplet data, the corresponding technology would be easily extended to more complicated star structures. Therefore, in the following discussions of our paper, we will focus on the processing of such triplet data, and use $X=\{x_1, x_2,\ldots, x_m\}$, $Y=\{y_1, y_2,\ldots, y_n\}$ and $Z=\{z_1, z_2,\ldots, z_t\}$ to represent the three types of objects. $Y$ is the central type that connects $X$ and $Z$. The distinct relations exist both between $X$ and $Y$, and between $Y$ and $Z$, but absent between $X$ and $Z$.

Recalling the discussion about the trivial extension of pair-wise co-clustering to the high-order case, although the relations among $X$, $Y$ and $Z$ could be represented by a tripartite graph, we can not use spectral cut of this graph to get the desirable co-clustering results. Instead, we model this problem as the consistent fusion of two pair-wise co-clustering sub-problems, with the constraint of the triplet structure. That is, we look for such two partitions for the sub-problems of $X$-$Y$ co-clustering and $Y$-$Z$ co-clustering, provided that each of them is not locally optimal, but their clustering results on the central type $Y$ are the same and the overall partitioning is globally optimal under a certain objective function. We call such partitions by consistent bipartite graph co-partitions. We proved in this paper that such kind of consistent partitions could be found by semi-definite programming (SDP). Then we tested the above ideas and the corresponding algorithms on both toy and real data. The results showed the feasibility and validity of our methods.

The rest of this paper is organized as follows. In Section 2 the background knowledge on graph-based clustering is introduced while the concept of consistency is proposed in Section 3. Then in Section 4 the method to solve the triplet co-clustering is described in details and the experimental results are discussed in Section 5. Concluding remarks and future work directions are listed in the last section.

## 2. SPECTRAL CLUSTERING

In this section, we will review some research works on spectral clustering, which serves as the foundation of our proposed concept of consistent bipartite graph co-partitioning.

### 2.1 Homogeneous Spectral Clustering

Spectral clustering [1][22] is a category of clustering algorithms based on spectral graph partitioning [18], which was proposed and well studied in the literature. To explain how this method works, we need to enumerate some basic knowledge in graph theory first.

A graph $G=(V, E)$ is composed by a set of vertices $V=\{1,2,\ldots,|V|\}$ and a set of edges $E=\{<i, j>|\ i, j\in V\}$, where $|V|$ represents the number of vertices. If using $E_{ij}$ to denote the weight of edge $<i,j>$, we can further define the adjacency matrix $M$ of the graph by

$$M_{ij}=\begin{cases}E_{ij}, & if <i,j>\in E\\ 0, & \text{otherwise}\end{cases}. \quad (1)$$

In the spectral graph partitioning methods, the vertices correspond to data objects, the edges correspond to the relationships among objects and the edge weights correspond to the strength of the relationships. Suppose the vertex set $V$ is partitioned into two subsets $V_1$ and $V_2$, then the corresponding *cut* might be defined as:

$$cut(V_1,V_2)=\sum_{i\in V_1, j\in V_2} M_{ij}. \quad (2)$$

The above definition can be easily extended to $k$ subsets:

$$cut(V_1,V_2,\ldots,V_k)=\sum_{\eta<\theta} cut(V_\eta,V_\theta). \quad (3)$$

Then the clustering is achieved by minimizing the *cut*. Usually, balanced clusters are more preferred, so some variations of the definition of *cut* were proposed and therefore different kinds of spectral clustering methods [6][12][22] were derived. For example, Ratio Cut [12] is achieved by balancing cluster sizes, while Normalized Cut [22] is attained by balancing cluster weights. Among these variations, Normalized Cut (or NCut) is one of the most popularly-used spectral clustering methods. Its objective function is shown in (4) where $e$ is the column vector with all its elements equal to 1:

$$\min \frac{q^T Lq}{q^T Dq}, \text{ subject to } q^T De=0, q\neq 0. \quad (4)$$

Here $D$ is a diagonal matrix with $D_{ii}=\sum_k E_{ik}$, and $L=D-M$ is called Laplacian matrix. $q$ is a column vector with $q_i= c_1$ if $i\in V_1$ and $q_i= -c_2$ if $i\in V_2$, where $c_1$ and $c_2$ are constants derived from $D$. By relaxing $q_i$ from discrete values to continuous values, it can be proved that the solution for (4) is the eigenvector corresponding

to the second smallest eigenvalue $\lambda_2$ of the following generalized eigenvalue problem [4][11][22]:

$$Lq = \lambda Dq \,. \tag{5}$$

Then we can obtain the desired clusters by running some routine clustering algorithms such as *k*-means [7] on this eigenvector $q$.

## 2.2 Bipartite Spectral Graph Partitioning

In order to use spectral graph partitioning to solve the pair-wise co-clustering problem, Dhillon [4] used the undirected bipartite graph[2] in Figure 2 to represent the relationship between the two types of heterogeneous objects. In this figure, squares and circles represent two types of objects $X = \{x_1, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ respectively, and the edges only exist between heterogeneous items. Then the bipartite graph can be represented by a triplet $G=(X, Y, E)$, where $E$ is a set of edges connecting vertices from different vertex sets, i.e., $E=\{<i, j> \mid i \in X, j \in Y\}$. If we further use $A$ to denote the inter-relation matrix in which $A_{ij}$ equals to $E_{ij}$, the adjacency matrix of the bipartite graph will be written as:

$$M = \begin{array}{c} \\ X \\ Y \end{array} \begin{array}{c} \begin{array}{cc} X & Y \end{array} \\ \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \end{array}, \tag{6}$$

where the vertices have been ordered such that the first $m$ vertices index the objects of $X$ while the last $n$ index the objects of $Y$.

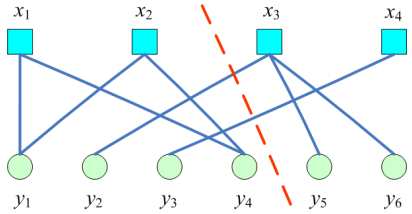


**Figure 2. The Bipartite Graph of *X* and *Y*.**

Suppose the dashed line in Figure 2 shows the very partition that minimizes (4), we will obtain two subsets $\{x_1,x_2,y_1,y_2,y_3,y_4\}$ and $\{x_3,x_4,y_5,y_6\}$. Therefore, the objects in $X$ are clustered into two subsets $\{x_1,x_2\}$ and $\{x_3,x_4\}$, while the objects in $Y$ are clustered into two subsets $\{y_1,y_2,y_3,y_4\}$ and $\{y_5,y_6\}$ simultaneously. To work out this very partition, we also need to solve a generalized eigenvalue problem like (5). Due to the bipartite property of the graph, after some trivial deduction, this problem can be converted to a singular value decomposition (SVD) [11] problem, which can be compute more efficiently. For the details of this algorithm, please refer to [4].

# 3. THE CONCEPT OF CONSISTENCY

After reviewing some background knowledge on spectral clustering, one natural question is whether this technology can be trivially extended to the high-order case. In this section, we will show that such an extension does not work as one expects. And then we will propose our concept of *consistency*, which models the high-order co-clustering problem as the fusion of pair-wise sub problems.

## 3.1 A Pretty Trap in Traditional Spectral Clustering

Consider the triplet data mentioned in the introduction. Inheriting the representations in Section 2, this triplet data can be pictured as a tripartite graph as shown in Figure 3 (for this specific example, the reasonable co-clustering results are labeled by the dashed line in the figure). If we use $A$ and $B$ to denote the inter-relationship matrices between $X$ and $Y$, and between $Y$ and $Z$ respectively, it is easy to derive the adjacency matrix for Figure 3:

$$M = \begin{array}{c} \\ X \\ Y \\ Z \end{array} \begin{array}{c} \begin{array}{ccc} X & Y & Z \end{array} \\ \begin{bmatrix} 0 & A & 0 \\ A^T & 0 & \alpha B \\ 0 & \alpha B^T & 0 \end{bmatrix} \end{array}, \tag{7}$$

where $\alpha$ is a weighting parameter, and the vertices have been ordered such that the first $m$ vertices index the objects of $X$, the next $n$ index the objects of $Y$ and the last $t$ index the objects of $Z$.

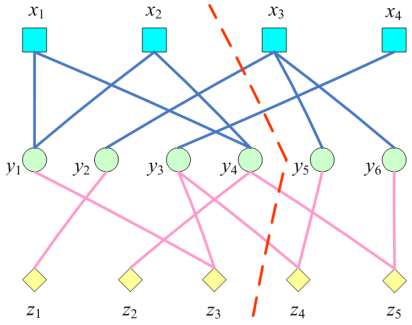


**Figure 3. The Tripartite Graph of *X*, *Y* and *Z*.**

Although it seems natural to partition the graph by working out the generalized eigenvalue problem corresponding to the adjacency matrix (7), we would like to point out that this idea does not always work as it seems. Actually, if we move the vertices of $X$ in Figure 3 to the side of the vertices of $Z$, the original tripartite graph will turn to be a bipartite graph as Figure 4 shows.

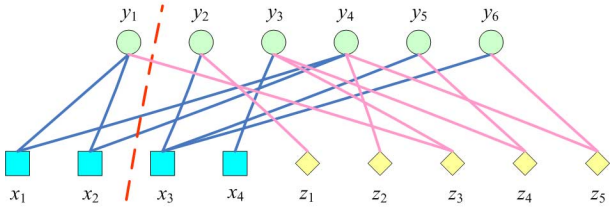


**Figure 4. The Bipartite Graph of *X*, *Y* and *Z*.**

Therefore, we are actually working on a $\{Y\}$-$\{X \,\&\, Z\}$ bipartite graph and have to distinguish the loss of cutting an $X$-$Y$ edge from the loss of cutting a $Y$-$Z$ edge since they contribute to the same loss function. However, these two kinds of edges are heterogeneous and might not be comparable. Although we could try to make them comparable by introducing the weighting parameter $\alpha$, however, it is non-trivial to choose a proper value for

[2] If the vertices of a graph can be decomposed into two disjoint subsets such that no two vertices within the same set are adjacent, the graph is named a bipartite graph.

$\alpha$ and we can not avoid the risk of assigning all vertices in $Z$ (or $X$) into one subset like the ill partitioning shown in Figure 4. In other words, analyzing the matrix $M$ in (7) by traditional spectral clustering method (we denote this approach by TSC for ease of reference) does not work as it is expected, and it is necessary for us to develop more advanced technology to handle high-order co-clustering.

## 3.2 Consistent Bipartite Graph Co-Partitioning (CBGC)

To tackle the aforementioned problem, we propose to treat the tripartite graph in Figure 3 as two bipartite graphs in Figure 2 and Figure 5 respectively, which share the central part of objects in $Y$. Then we transform the original high-order problem to the fusion of the pair-wise co-clustering problems over these two bipartite graphs.

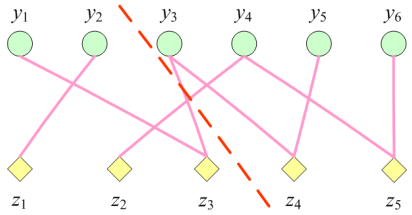


**Figure 5. The Bipartite Graph of *Y* and *Z*.**

However, if we conduct bipartite spectral graph partitioning on Figure 2 and 5 independently, it will have a great probability that the partitioning scheme for $Y$ is different in the two solutions. In other words, the two locally optimal partitioning schemes in $Y$ do not match in most cases. This is not what we want. Actually, we are looking for such two partitions for Figure 2 and 5, provided that each of them is not locally optimal, but their clustering results on the central type $Y$ are the same, and the overall partitioning is globally optimal under a certain objective function. We call it by *consistent bipartite graph co-partitioning* (CBGC).

So far, the aforementioned concept of CBGC is very generic. To make it computable, we will give a specific objective function and discuss how to optimize it efficiently. Note that, in this paper, we will only focus on a partition of two clusters, where all types of objects will be simultaneously clustered into two groups respectively. For this purpose, we let $x$, $y$, $z$ act as the indicating column vectors of $m$, $n$, $t$ dimensions for the objects in $X$, $Y$, $Z$ respectively. We denote $q=(x, y)^T$ and $p=(y, z)^T$ as the indicating vectors for the two local bipartite graphs, and denote $D^{(1)}$, $D^{(2)}$, $L^{(1)}$ and $L^{(2)}$ as the diagonal matrices and Laplacian matrices for the adjacent matrices $A$ and $B$. Then we mathematically model the consistent co-partitioning problem as follows,

$$\begin{cases} \min \beta \dfrac{q^T L^{(1)} q}{q^T D^{(1)} q} + (1-\beta)\dfrac{p^T L^{(2)} p}{p^T D^{(2)} p} \\ \text{subject to } q^T D^{(1)} e = 0,\ q \neq 0 \\ \qquad\qquad p^T D^{(2)} e = 0,\ p \neq 0 \\ \qquad\qquad 0 < \beta < 1 \end{cases}, \tag{8}$$

where $\beta$ is a weighting parameter to balance which local graph we trust more. We can see that the above additive objective function commendably reflects the concept of consistency, and the two constraints might avoid the awkward situation in Figure 4.

# 4. SOLVING CBGC BY SEMI-DEFINITE PROGRAMMING

In this section we will propose an algorithm to compute the solution of the optimization problem defined in Section 3.2. The core idea is to convert it to a semi-definite programming (SDP) problem so that it can be computed efficiently.

For this purpose, we set $\omega=(x, y, z)^T$ to be a union indicating vector of $s=m+n+t$ dimensions, and extend the matrices $L^{(1)}$,$L^{(2)}$,$D^{(1)}$ and $D^{(2)}$ to adapt the length of $\omega$:

$$\Gamma_1 = \begin{bmatrix} L^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{s\times s},\ \Gamma_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & L^{(2)} \end{bmatrix}_{s\times s}, \tag{9}$$

$$\Pi_1 = \begin{bmatrix} D^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{s\times s},\ \Pi_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & D^{(2)} \end{bmatrix}_{s\times s}, \tag{10}$$

where the **0**'s are matrix blocks with all the elements equal to zero. Then accordingly (8) can be rewritten as:

$$\begin{cases} \min\left(\beta\dfrac{\omega^T\Gamma_1\omega}{\omega^T\Pi_1\omega} + (1-\beta)\dfrac{\omega^T\Gamma_2\omega}{\omega^T\Pi_2\omega}\right) \\ \text{subject to } \omega^T\Pi_1 e = 0 \\ \qquad\qquad \omega^T\Pi_2 e = 0 \\ \qquad\qquad \omega \neq 0,\ 0 < \beta < 1 \end{cases}. \tag{11}$$

Problem (11) is a typical sum-of-ratios quadratic fractional programming problem [8], which is hard and complicated to solve although there has been some branch-and-bound algorithms [2]. To avoid solving this fractional programming problem, we use a familiar skill in spectral clustering to simplify it: by fixing the values of the denominators in (11) to $e^T\Pi_1 e$ and $e^T\Pi_2 e$ respectively, we have:

$$\begin{cases} \min \omega^T\Gamma\omega \\ \text{subject to } \omega^T\Pi_1\omega = e^T\Pi_1 e \\ \qquad\qquad \omega^T\Pi_2\omega = e^T\Pi_2 e \\ \qquad\qquad \omega^T\Pi_1 e = 0 \\ \qquad\qquad \omega^T\Pi_2 e = 0 \end{cases}, \tag{12}$$

$$\text{where}\quad \Gamma = \frac{\beta}{e^T\Pi_1 e}\Gamma_1 + \frac{1-\beta}{e^T\Pi_2 e}\Gamma_2,\ 0 < \beta < 1. \tag{13}$$

Optimization problem (12) turns to be a quadratically constrained quadratic programming (QCQP) [3] problem and it is not difficult to verify that the constraints are all convex because matrices $\Pi_1$ and $\Pi_2$ are both positive semi-definite. As we know, convex QCQP problem can be cast in the form of a semi-definite programming problem (SDP) [3] for efficient computation.

SDP is an optimization problem with the form as below:

$$\begin{cases} \min C \bullet W \\ \text{subject to } A_i \bullet W = b_i, i = 1,...,k \\ \qquad\qquad W \succeq 0 \end{cases}, \tag{14}$$

where $C$ is a symmetric coefficient matrix and $W$ is a symmetric parameter matrix; the symbol $\succeq 0$ means positive semi-definite; $A_i$ (and $b_i$), $i$=1,…,$k$ are coefficient matrices (and vectors) for the constraints; the matrix inner-product is defined as:

$$C \bullet W = \sum_{i,j} C_{ij} W_{ij} \ . \tag{15}$$

As it turns out, QCQP can be reformed as a SDP by relaxing the product terms $\omega_i\omega_j$ to an element $\Omega_{ij}$ of a symmetric matrix $\Omega$. To show this, we begin with the following elementary proposition.

**PROPOSITION**. *Given a vector* $\omega \in R^k$ *and a matrix* $\Omega \in R^{k\times k}$, *then* $\begin{bmatrix} 1 & \omega^T \\ \omega & \Omega \end{bmatrix}$ *is positive semi-definite if and only if* $\Omega - \omega\omega^T$ *is positive semi-definite.*

**Proof.** This is a standard result from linear algebra [11].□

Using this proposition, it is straightforward to show that the QCQP in (12) is equivalent to the following SDP:

$$\begin{cases} \min_{\omega,\Omega} \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & \Gamma \end{bmatrix} \bullet \begin{bmatrix} 1 & \omega^T \\ \omega & \Omega \end{bmatrix} \\ \text{subject to } \begin{bmatrix} -e^T\Pi_1 e & \mathbf{0} \\ \mathbf{0} & \Pi_1 \end{bmatrix} \bullet \begin{bmatrix} 1 & \omega^T \\ \omega & \Omega \end{bmatrix} = 0 \\ \qquad \begin{bmatrix} -e^T\Pi_2 e & \mathbf{0} \\ \mathbf{0} & \Pi_2 \end{bmatrix} \bullet \begin{bmatrix} 1 & \omega^T \\ \omega & \Omega \end{bmatrix} = 0 \\ \qquad \begin{bmatrix} 0 & e^T\Pi_1/2 \\ \Pi_1 e/2 & \mathbf{0} \end{bmatrix} \bullet \begin{bmatrix} 1 & \omega^T \\ \omega & \Omega \end{bmatrix} = 0 \\ \qquad \begin{bmatrix} 0 & e^T\Pi_2/2 \\ \Pi_2 e/2 & \mathbf{0} \end{bmatrix} \bullet \begin{bmatrix} 1 & \omega^T \\ \omega & \Omega \end{bmatrix} = 0 \\ \qquad \begin{bmatrix} 1 & \omega^T \\ \omega & \Omega \end{bmatrix} \succeq 0 \end{cases} \tag{16}$$

As it has been proved that the SDP relaxation of a QCQP may produce an approximation to the original problem with a good error bound [21], we further ignore the constraints of $\Omega = \omega_i\omega_j$ and get the following relaxation:

$$\begin{cases} \min_W \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & \Gamma \end{bmatrix} \bullet W \\ \text{subject to } \begin{bmatrix} -e^T\Pi_1 e & \mathbf{0} \\ \mathbf{0} & \Pi_1 \end{bmatrix} \bullet W = 0 \\ \qquad \begin{bmatrix} -e^T\Pi_2 e & \mathbf{0} \\ \mathbf{0} & \Pi_2 \end{bmatrix} \bullet W = 0 \\ \qquad \begin{bmatrix} 0 & e^T\Pi_1/2 \\ \Pi_1 e/2 & \mathbf{0} \end{bmatrix} \bullet W = 0 \\ \qquad \begin{bmatrix} 0 & e^T\Pi_2/2 \\ \Pi_2 e/2 & \mathbf{0} \end{bmatrix} \bullet W = 0 \\ \qquad \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \bullet W = 1, \\ \qquad \begin{bmatrix} 0 & e \\ e & \mathbf{0} \end{bmatrix} \bullet W = \theta_1, \\ \qquad \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & E \end{bmatrix} \bullet W = \theta_2 \\ \qquad W \succeq 0 \end{cases} \tag{17}$$

where $E$ is a matrix block with all the elements equal to one; the constraint $\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \bullet W = 1$ guarantees $W_{11}$=1, and the next two constrains are bound controllers with some constants $\theta_1$ and $\theta_2$.

Up to now, we have got a standard form of SDP. The first column of $W$ (except $W_{11}$) can be regarded as the representation of $\omega$. SDP is a hot research field [21] in recent years, and many fast iterative algorithms have been designed to solve it [13][16][17]. For example, an interior-point method SDPA [20] was implemented in [9] for solving the standard form SDP and its dual problem. We could use it to compute an efficient solution to the optimization problem (17).

To summarize, our algorithm to solve the co-clustering of triplet data can be listed as below. For ease of reference, we also use CBGC to abbreviate it in the future discussions.

**The CBGC Algorithm**

1. Set the parameters $\beta$, $\theta_1$ and $\theta_2$.
2. Given the inter-relation matrices $A$ and $B$, form the corresponding diagonal matrices and Laplacian matrices $D^{(1)}$, $D^{(2)}$, $L^{(1)}$ and $L^{(2)}$.
3. Extend $D^{(1)}$, $D^{(2)}$, $L^{(1)}$ and $L^{(2)}$ to $\Pi_1$, $\Pi_2$, $\Gamma_1$ and $\Gamma_2$, and form $\Gamma$, such that the coefficient matrices in SDP (17) can be computed.
4. Solve (17) by a certain iterative algorithm such as SDPA.
5. Extract $\omega$ from $W$ and regard it as the embedding vector of the heterogeneous objects.
6. Run the $k$-means algorithm on $\omega$ to obtain the desired partitioning of the heterogeneous objects.

Although when explaining the concept and the algorithm of CBGC, we take the triplet heterogeneous data co-clustering for example, here we want to point out that they can be easily generalized to solve the co-clustering of star-structured $k$-partite heterogeneous objects. In such case, we only need to refine the optimization problem (8) to the following form:

$$\begin{cases} \min \sum_{i=1}^{k-1} \beta_i \dfrac{q_i^T L^{(i)} q_i}{q_i^T D^{(i)} q_i} \\ \text{subject to } \; q_i^T D^{(i)} e = 0,\ q_i \neq 0, i = 1,\ldots,k-1 \\ \qquad\qquad \sum_{i=1}^{k-1} \beta_i = 1,\ 0 < \beta_i < 1 \end{cases} \tag{18}$$

where $q_i$, $i$=1,..,$k$-1 are indicating vectors for the local bipartite graphs. Once again, (18) can be re-organized as a QCQP and further solved as a SDP.

# 5. EXPERIMENTAL EVALUATION

In this section, we evaluated the effectiveness of our proposed consistency concept and the corresponding SDP-based CBGC algorithm. For this purpose, we conducted three experiments. The first one was a toy problem to show that the proposed CBGC algorithm could avoid the ill partitioning situation in Figure 4 to many extents. The second experiment was another toy to show that our method could get a trade off partitioning on the central type of objects under the concept of consistency. The last experiment was implemented on a real data set to confirm our conclusions.

## 5.1 Toy Problem I

In the first toy problem, the set sizes of $X$, $Y$ and $Z$ were 3, 8 and 6 respectively, and the inner-relationships among objects were shown in Figure 6. Suppose the edge weights were all assigned with 1, then it is not difficult to get that the partitioning scheme illuminated by the dashed lines in Figure 6 might be what we wanted. On this toy data, we compared the clustering results produced by the traditional spectral clustering method (TSC) and our CBGC algorithm.

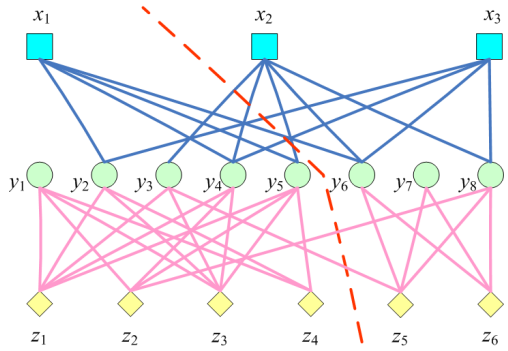


**Figure 6. The Tripartite Graph of Toy Problem I.**

### 5.1.1 *Clustering Results of TSC*

In TSC, we treated Figure 6 as an ordinary graph and worked out the generalized eigenvalue problem corresponding to the adjacency matrix like (7). We tuned $\alpha$ in a large range and plotted the embeddings when $\alpha$ = 0.01, 1, and 100 in Figure 7, 8 and 9. The vertical axis reflected the corresponding embedding values, and the colors of the points showed the $k$-means clustering results. As Figure 6 was actually a $\{Y\}$-$\{X$ & $Z\}$ bipartite graph, the embedding values of $X$ and $Z$ were mixed together in the results. For example, the first 3 points in Figure 7(a) represented $X$ (from $x_1$ to $x_3$), while the next 6 points represented $Z$ (from $z_1$ to $z_6$); the embedding of $Y$ (from $y_1$ to $y_8$) was shown in Figure 7(b).

From Figure 7 to 9 and many results that were not listed in this paper, we could find that TSC failed in partitioning $X$ no matter what value $\alpha$ took: all vertices in $X$ and a part of vertices in $Z$ were always clustered together while the rest vertices in $Z$ made another cluster. This told us that in some cases, the ill partitioning results as mentioned in Section 3.1 could hardly be avoided if we treat the relationship among heterogeneous data as a homogeneous graph.

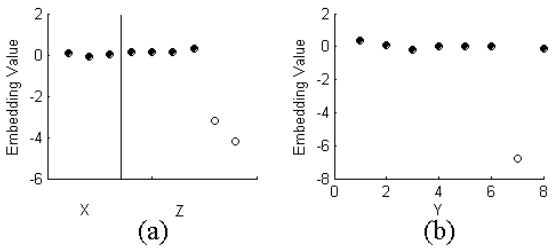


**Figure 7. The Embeddings Produced by TSC ($\alpha$ = 0.01).**

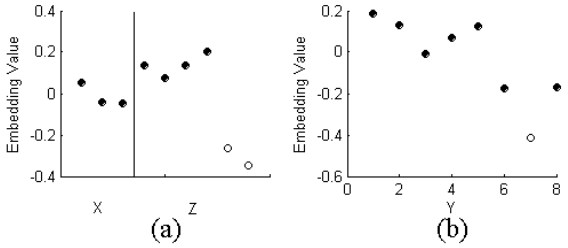


**Figure 8. The Embeddings Produced by TSC ($\alpha$ = 1).**

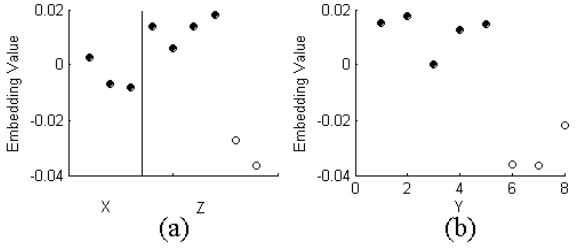


**Figure 9. The Embeddings Produced by TSC ($\alpha$ = 100).**

### 5.1.2 *Clustering Results of CBGC*

In this subsection, we ran CBGC also on the toy data as shown in Figure 6. Here, we use $A$ to denote the adjacency matrix of the $X$-$Y$ bipartite graph and use $B$ to denote that of the $Y$-$Z$ bipartite graph. As the embedding values of $X$, $Y$ and $Z$ were extracted from the resulting matrix $W$ of SDP, they were mixed together in one vector $\omega$. Specifically, when we set $\beta$=0.3 and $\theta_1$=$\theta_2$=1, we got the embeddings in this vector as shown in Figure 10. As can be seen, CBGC successfully avoided the ill partitioning and the co-

clustering results of the three heterogeneous objects were quite accordant with our desire.

Then we turned the value of $\beta$ in the interval [0, 1]. We found that the clustering results changed along with the changing of $\beta$. For example, when setting $\beta$=0.1, we got the results in Figure 11, in which the partitioning on $X$ was different from in Figure 10 because matrix $B$ had much more influence than $A$. When setting $\beta$=0.9, matrix $A$ became dominant so that $y_7$ and $z_6$ were picked out from the whole set (see Figure 12). This is also reasonable because the $X$-$Y$ bipartite graph was actually unconnected. If we further change the value of $\beta$ to very close to 0 or 1, the partitioning on $X$ or $Z$ will fail because the co-clustering actually degrade to be working on only one bipartite graph.

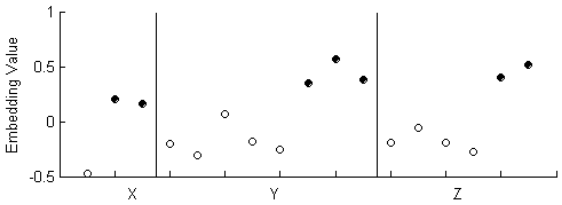


**Figure 10. The Embeddings Produced by CBGC ($\beta$=0.3).**

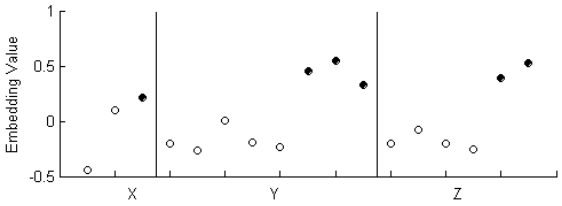


**Figure 11. The Embeddings Produced by CBGC ($\beta$=0.1).**

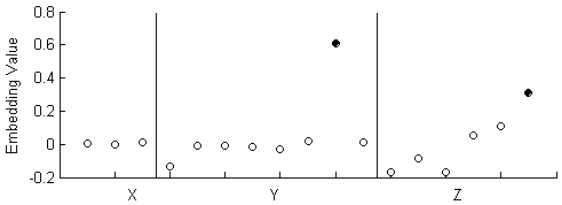


**Figure 12. The Embeddings Produced by CBGC ($\beta$=0.9).**

To summarize, this experiment showed that CBGC can avoid the case of ill partitioning to many extents, and tuning $\beta$ is an effective way to trade-off between the two bipartite graphs.

## 5.2 Toy Problem II

In the second toy problem, the set sizes of $X$, $Y$ and $Z$ were 16, 20 and 21 respectively, and the two inter-relationship matrices $A$ and $B$ were shown in Figure 13 and 14, where the size of the circle at $(i,j)$ illuminated the corresponding edge weight (the quantization range is 0~9).

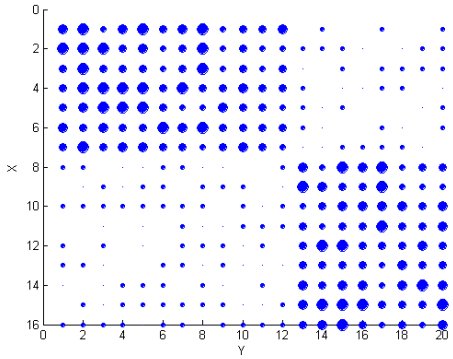


**Figure 13. The Inter-relation Matrix $A$ for Toy Problem II.**

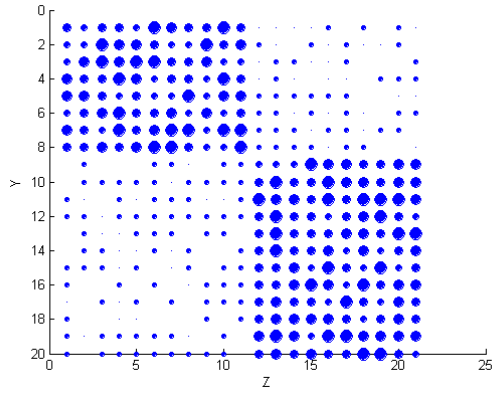


**Figure 14. The Inter-relation Matrix $B$ for Toy Problem II.**

From matrix $A$, we could find that the reasonable co-clustering results of the $X$-$Y$ bipartite graph should be $\{x_1,\ldots,x_7\}$ vs $\{x_8,\ldots,x_{16}\}$ and $\{y_1,\ldots y_{12}\}$ vs $\{y_{13},\ldots,y_{20}\}$. From matrix $B$, we could find that the co-clustering results of the $Y$-$Z$ bipartite graph should be $\{y_1,\ldots,y_8\}$ vs $\{y_9,\ldots y_{20}\}$ and $\{z_1,\ldots,z_{11}\}$ vs $\{z_{12},\ldots,z_{21}\}$. It was clear that the partitioning scheme for $Y$ was different in the two bipartite graphs. In such a situation, it is an interesting question what kind of co-clustering results our CBGC algorithm will produce.

To get a comprehensive answer to this question, we tuned the parameter $\beta$ in CBGC from 0 to 1 with a step of 0.2, and got the embedding values as shown in Figure 15 to 20. Here the parameters $\theta_1$ and $\theta_2$ were both set to 1. From these figures, we could find that when $\beta$=0, matrix $A$ was suppressed to zero and the partitioning on $Y$ and $Z$ were completely accordant with what matrix $B$ showed. When $\beta$=0.6, the partitioning scheme on $Y$ was a tradeoff between the two locally optimal partitioning schemes of the two bipartite graphs. This well showed the core idea of our consistency concept. When $\beta$ went up to 1, matrix $B$ vanished and the partitioning on $X$ and $Y$ were in accordance with what matrix $A$ showed. Besides these three typical cases, $\beta$=0.2, 0.4, 0.8 just showed how the partitioning schemes transited from one case to another.

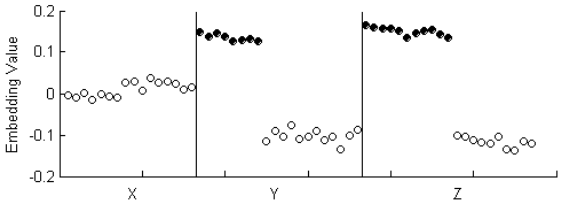


**Figure 15. The Embeddings Produced by CBGC ($\beta$=0.0).**

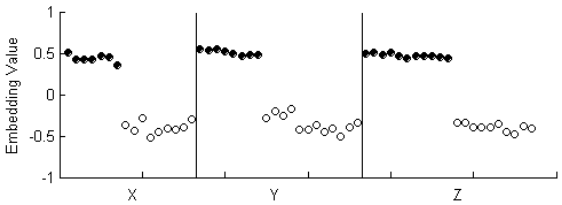


**Figure 16. The Embeddings Produced by CBGC ($\beta$=0.2).**

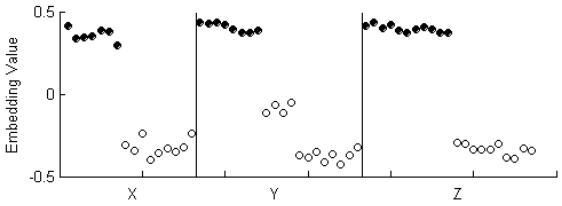


**Figure 17. The Embeddings Produced by CBGC ($\beta$=0.4).**

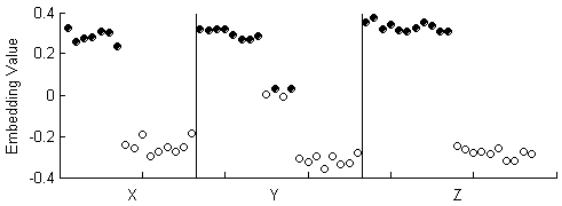


**Figure 18. The Embeddings Produced by CBGC ($\beta$=0.6).**

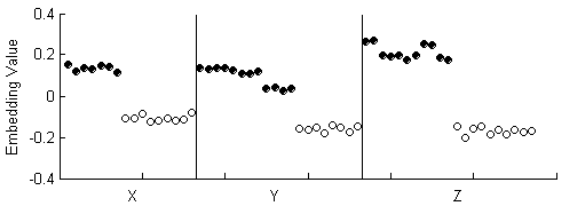


**Figure 19. The Embeddings Produced by CBGC ($\beta$=0.8).**

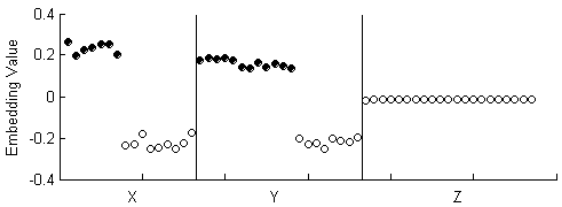


**Figure 20. The Embeddings Produced by CBGC ($\beta$=1.0).**

## 5.3 Real Data

In the field of text categorization, hierarchical taxonomy classification is widely used to get better trade-off between effectiveness and efficiency compared with flat taxonomy classification. Unfortunately, many data sets are not explicitly organized in hierarchical forms. To take advantages of hierarchical classification, people have to mine a hierarchical taxonomy from the data set. We can see that the relationship between categories, documents and terms is just a star structure, so we modeled it by a tripartite graph and solved it by the CBGC algorithm. Then the category clusters could be used to generate a hierarchical taxonomy. In this section, we would like to show that the proposed algorithm could output reasonable category clusters for hierarchical taxonomy building, while the building process is omitted, for the details of which, please refer to [10].

The real data used in our experiments were sampled from the dataset of 20-newsgroups[3] which contains about 20,000 articles from 20 newsgroups. We picked five categories (see Table 1) and randomly select 30 articles for each category. Then we used the technique as described in the Appendix to build the category-by-document matrix $A$ and used term frequency to build the document-by-term matrix $B$. Here we carried out feature selection according to [24] for terms so that only 533 terms were reserved. It could be easily seen from the category names listed in Table 1 that the expected co-clustering results on categories should be $\{C_1, C_2\}$ and $\{C_3, C_4, C_5\}$, and the documents should be clustered according to the categories they belong to.

**Table 1. A real data set sampled from 20-newsgroups.**

| Category Name | Notation |
|---|---|
| *rec.sport.baseball* | $C_1$ |
| *rec.sport.hockey* | $C_2$ |
| *talk.politics.guns* | $C_3$ |
| *talk.politics.mideast* | $C_4$ |
| *talk.politics.misc* | $C_5$ |

### *5.3.1 Clustering Results of TSC*

We tuned different values of $\alpha$ to see the performance of the TSC algorithm. From Table 2 we can see that the results were disappointing no matter how this parameter was set. In fact, when $\alpha$ was small, the algorithm degenerated to co-cluster on a category-document bipartite graph which is unconnected in this data set. Otherwise, all categories were assigned to one cluster.

**Table 2. Performance under different values of $\alpha$.**

| $\alpha$ | Cluster 1 | Cluster 2 |
|---|---|---|
| 1e-10 | $C_1, C_2, C_4, C_5$ | $C_3$ |
| 0.01 | $C_1, C_2, C_4, C_5$ | $C_3$ |
| 0.1 | $C_1, C_2, C_3, C_4, C_5$ | *null* |
| 1 | $C_1, C_2, C_3, C_4, C_5$ | *null* |
| 10 | $C_1, C_2, C_3, C_4, C_5$ | *null* |
| 100 | $C_1, C_2, C_3, C_4, C_5$ | *null* |
| 1e10 | $C_1, C_2, C_3, C_4, C_5$ | *null* |

[3] http://people.csail.mit.edu/~jrennie/20Newsgroups

### *5.3.2 Clustering Results of CBGC*

Then we ran CBGC on the same real data with $\theta_1=\theta_2=1$. The results under different settings of $\beta$ were shown in Figure 21 to 23. As the total number of categories, documents and terms was too large, we plotted the embeddings of terms in a separate sub-figure to show the results distinctly, although the embedding values of the three heterogeneous objects were co-clustered together.

From Figure 21, we found that when $\beta$=0.5, the first two categories ($C_1$ and $C_2$) were clustered together with their documents (the first 60 documents) and a part of terms (the dark-colored points), while the rest categories ($C_3$, $C_4$ and $C_5$), documents (the last 90 documents) and terms (the white-colored points) were grouped into another cluster. When $\beta$ was set to 0 and 1, the results (Figure 22 and 23) looked similar to the degraded cases as shown in Figure 15 and 20. These results confirmed from a practical point of view the advantages of our consistent bipartite graph co-partitioning over analyzing the category-document and document-term bipartite graphs separately.

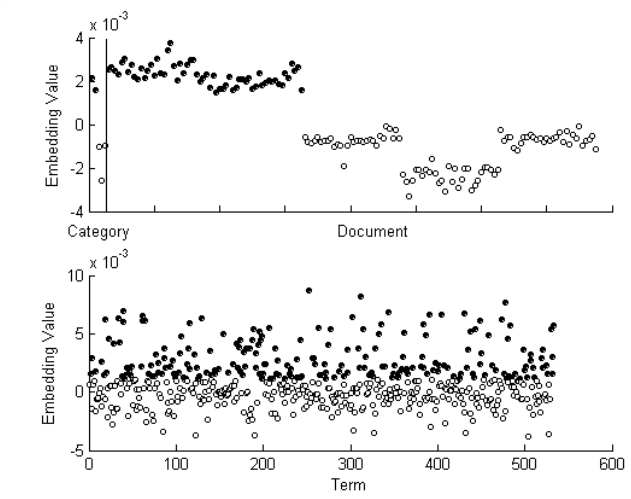


**Figure 21. The Embeddings Produced by CBGC ($\beta$=0.5).**

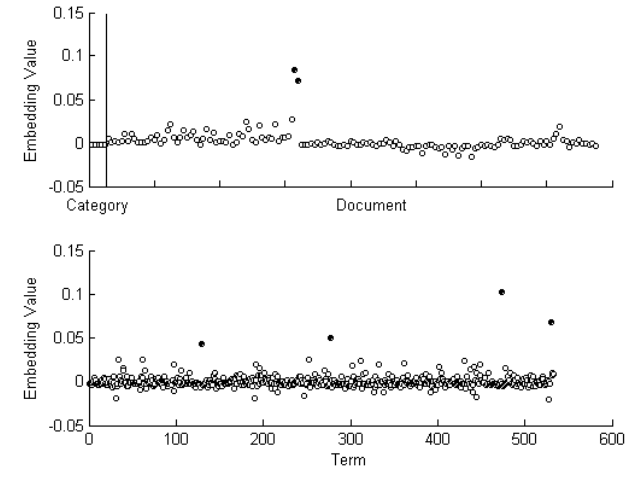


**Figure 22. The Embeddings Produced by CBGC ($\beta$=0.0).**

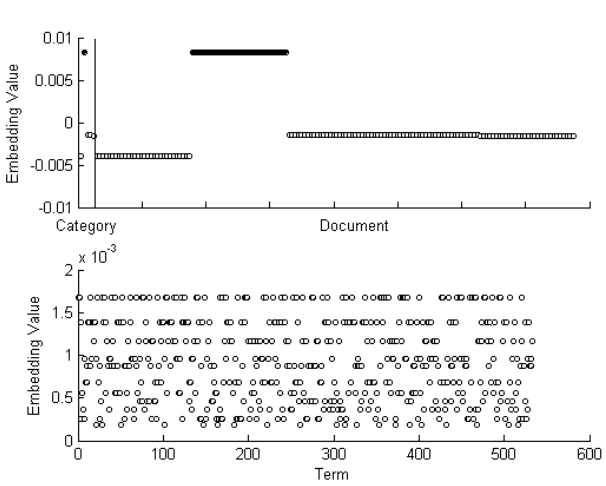


**Figure 23. The Embeddings Produced by CBGC ($\beta$=1.0).**

To summarize, our experiments on both toy and real data showed that our concept of consistent bipartite graph co-partitioning can well handle the star-structured high-order heterogeneous data co-clustering, and greatly outperform the traditional spectral clustering either on separate bipartite graphs or on the tripartite graph.

## 6. CONCLUSIONS AND FUTURE WORK

In this paper, we used a *k*-partite graph to represent the star-structured inter-relationships among high-order heterogeneous objects, and proposed the concept of consistent bipartite graph co-partitioning to get the co-clustering of these objects simultaneously. Then we proved our desired consistent co-clustering can be achieved by optimizing a certain objective function based on semi-definite programming. Experiments on both toy and real data showed that our approach worked effectively and efficiently.

For the future work, we plan to investigate the following issues:

1. So far, in this paper we only discussed the case of a partition of two clusters. Although our unreported experiments showed that the indicating vector produced by SDP also embeds rich information for the *k*-partitioning cases, currently its theoretical effectiveness still needs proving.
2. We will put effort on developing a fast and simple technique to choose the parameters ($\beta$, $\theta_1$ and $\theta_2$) automatically, for when dealing with *k*-partite graphs, there are *k* parameters of $\beta$ to estimate, in the case of which the task of parameter estimation might be burdensome. We have found the work in [15] quite useful in the guidance of choosing parameters and our corresponding work is ongoing.
3. We are willing to implement the proposed algorithm on larger datasets to discuss the scalability issues and on other applications such as image clustering to see the performance.
4. We will study how the CBGC algorithm will perform if we relax the consistent partitioning on the central type of objects from exactly the same to almost the same (i.e. soft consistency).
5. We will further explore whether there are any more reasonable objective functions, and whether it is possible to get a close-form solution for them.

6. We also plan to apply our method to extend other algorithms such as information-theoretic co-clustering [5] to handle high-order heterogeneous data.

## 7. ACKNOWLEDGMENTS

We would like to thank Professor Kurt M. Anstreicher for his great generosity and enthusiasm in helping us nail down some facts on semi-definite programming. We should also thank Tao Qin, Guang Feng and Huai-Yuan Yang for their constructive suggestions on this work. Dr. Tie-Yan Liu was the corresponding author of this paper.

## 8. REFERENCES

[1] Bach, F.R., and Jordan, M.I. Learning spectral clustering. Neural Info. Processing Systems 16 (NIPS 2003), 2003.

[2] Benson, H.P. Global Optimization Algorithm for the Nonlinear Sum of Ratios Problem. Journal of Optimization Theory and Applications: Vol. 112, No. 1, pp. 1–29, January 2002.

[3] Boyd, S., and Vandenberghe, L. Convex Optimization. Cambridge University Press, 2004.

[4] Dhillon, I.S. Co-clustering documents and words using bipartite spectral graph partitioning. In KDD'01, 2001.

[5] Dhillon, I.S., Mallela, S., and Modha, D.S. Information-Theoretic co-clustering. SIGKDD '03, 2003.

[6] Ding, C., He, X., Zha, H., Gu, M., and Simon, H. A min-max cut algorithm for graph partitioning and data clustering. Proc. IEEE Int'l Conf. Data Mining, 2001.

[7] Duda, R.O., Hart, P.E., and Stork, D.G. Pattern classification, Second Edition. John Wiley & Sons Inc. 2001.

[8] Frenk, J.B.G., and Schaible, S. Fractional Programming. ERIM Report Series Reference No. ERS-2004-074-LIS. http://ssrn.com/abstract=595012.

[9] Fujisawa, K., Fukuda, M., Kojima, M., and Nakata, K. Numerical evaluation of the SDPA (SemiDefinite Programming Algorithm). High Performance Optimization, Kluwer Academic Press, 267-301, 2000.

[10] Gao, B., Liu, T., Cheng, Q., Feng, G., Qin, T., and Ma, W. Hierarchical Taxonomy Preparation for Text Categorization Using Consistent Bipartite Spectral Graph Co-partitioning. Accepted for publication, IEEE Transactions on Knowledge and Data Engineering, Special Issue on Data Preparation, 2005.

[11] Golub, G.H., and Loan, C.F.V. Matrix computations. Johns Hopkins University Press, 3rd edition, 1996.

[12] Hagen, L., and Kahng, A.B. New spectral methods for ratio cut partitioning and clustering. IEEE. Trans. on Computed Aided Desgin, 11:1074-1085, 1992.

[13] Klerk, E. Aspects of Semidefinite Programming: Interior Point Algorithms and Selected Applications. Applied Optimization Series, Volume 65. Kluwer Academic Publishers, March 2002, 300 pp., ISBN 1-4020-0547-4.

[14] Kluger, Y., Basri, R., Chang, J.T., and Gerstein, M. Spectral biclustering of microarray cancer data: co-clustering genes and conditions. Genome Res., Apr 2003; 13: 703 - 716.

[15] Modha, D.S., and Spangler, W.S. Feature Weighting in *k*-Means Clustering. Machine Learning, Volume 52, Issue 3, Sep 2003, Pages 217-237.

[16] Monteiro, R.D.C. First- and Second-Order Methods for Semidefinite Programming. Georgia Tech, January 2003.

[17] Pardalos, P.M. and Wolkowicz, H. Topics in Semidefinite and Interior Point Methods. Fields Institute Communications 18, AMS, Providence, Rhode Island, 1998.

[18] Pothen, A., Simon, H.D., and Liou, K.P. Partitioning sparse matrices with eigenvectors of graph. SIAM Journal of Matrix Anal. Appl., 11:430-452, 1990.

[19] Qiu, G. Image and Feature Co-clustering. ICPR (4) 2004: 991-994.

[20] SDPA Online for your future. http://grid.r.dendai.ac.jp/sdpa/.

[21] Semidefinite Programming. http://www-user.tu-chemnitz.de/~helmberg/semidef.html.

[22] Shi, J., and Malik, J. Normalized cuts and image segmentation. IEEE. Trans. on Pattern Analysis and Machine Intelligence, 22:888-905, 2000.

[23] Wang, J., Zeng, H., Chen, Z., Lu, H., Tao, L., and Ma, W. ReCoM: reinforcement clustering of multi-type interrelated data objects. Proceedings of ACM SIGIR'03, 2003, Toronto, Canada.

[24] Yang, Y., and Pedersen J.P. A Comparative Study on Feature Selection in Text Categorization. Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97), 1997, pp412-420.

[25] Zeng, H., Chen, Z., and Ma, W. A Unified Framework for Clustering Heterogeneous Web Objects. In Proc. 3rd WISE 2002, 12-14 December, Singapore, IEEE Computer Society (2002) 161–172.

[26] Zha, H., Ding, C., and Gu, M. Bipartite graph partitioning and data clustering. In proceedings of CIKM'01, 2001.

## 9. APPENDIX

In this appendix we would like to illuminate how we built the category-by-document matrix $A$ in Section 5.3.

Suppose $D=\{d_1,d_2,\ldots,d_n\}$ denotes the documents in the dataset for text categorization and $T=\{w_1,w_2,\ldots,w_t\}$ denotes the terms, then each document $d_i$ in $D$ can be represented by a $t$-dimensional vector $d_i=\{x_{i1},x_{i2},\ldots,x_{it}\}$, where $x_{ij}$ is the term frequency of term $w_j$ in document $d_i$. Furthermore, each document is assigned a category label from the set $C=\{c_1,c_2,\ldots,c_m\}$, where $m$ is the total number of categories.

The category-by-document matrix $A$ can be easily built according to the information from the corpus. In this matrix, rows correspond to categories and columns to documents. Each element $A_{ij}$ indicates the correlation between document $d_j$ and category $c_i$. If document $d_j$ belongs to $k$ categories $c_1$, $c_2$, … , $c_k$, the weights $A_{1j}$, $A_{2j}$, … , $A_{kj}$ are set to $1/k$, and the other elements of the $j$-th column of matrix $A$ are set to zero.

After a series of processing as described above, the category-by-document matrix $A$ can be easily obtained.□