Graph Theoretical Framework for Simultaneously Integrating Visual and Textual Features for Efficient Web Image Clustering

Manjeet Rege Ming Dong Jing Hua
Machine Vision Pattern Recognition Lab Graphics and Imaging Lab
Department of Computer Science, Wayne State University
Detroit, MI 48202, USA
{rege, mdong, jinghua}@wayne.edu

ABSTRACT

With the explosive growth of Web and the recent development in digital media technology, the number of images on the Web has grown tremendously. Consequently, Web image clustering has emerged as an important application. Some of the initial efforts along this direction revolved around clustering Web images based on the visual features of images or textual features by making use of the text surrounding the images. However, not much work has been done in using multimodal information for clustering Web images. In this paper, we propose a graph theoretical framework for simultaneously integrating visual and textual features for efficient Web image clustering. Specifically, we model visual features, images and words from surrounding text using a tripartite graph. Partitioning this graph leads to clustering of the Web images. Although, graph partitioning approach has been adopted before, the main contribution of this work lies in a new algorithm that we propose - Consistent Isoperimetric High-order Co-clustering (CIHC), for partitioning the tripartite graph. Computationally, CIHC is very quick as it requires a simple solution to a sparse system of linear equations. Our theoretical analysis and extensive experiments performed on real Web images demonstrate the performance of CIHC in terms of the quality, efficiency and scalability in partitioning the visual feature-image-word tripartite graph.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Experimentation, Theory.

1. INTRODUCTION

Over the last decade, the Web has evolved from its initial days of infancy into an unstructured database that we know of today. The initial search engines applied the text information retrieval model wherein a Web document was retrieved based on the relevance matching between the user provided query and the words appearing in the document. However, with the recent development in the field of digital media technology which has resulted in the generation of a huge number of images rapidly, efficient Web image search and browsing have become an important application. Consequently, Web image clustering has drawn significant attention in the research community recently. For example, properly grouped Web images can provide a very neat bird's eye view of the retrieved images to the user.

Much of the earlier efforts on image clustering were solely based on low-level visual features of images [14, 25]. To cluster images, visual features such as color histogram or wavelet-based [13] texture extracted from images were subjected to traditional data clustering algorithms [17]. The major drawback in this approach is that the state-of-the-art visual features are unable to represent the image content on a semantic level. As a result, image clustering suffers from the semantic gap between visual features and high-level semantic concepts. A low-tech and a naive solution adopted by search engines to overcome this problem to a certain extent has been to treat image clustering as a text clustering problem. Web images are represented using textual features in terms of the surrounding texts and captions. Images clustered based on these textual features are then retrieved accordingly. But, since images are not actually text documents, this approach is hardly a solution to the problem at hand.

A better approach is to incorporate both visual and textual features together for efficient Web image clustering. In [3], a single index vector of an image is formed by combining textual and visual statistics. Textual statistics are captured in a vector form using latent semantic indexing (LSI) [5] based on text in the containing HTML document, while visual statistics are also stored in a vector form using color and orientation histograms. A similar approach was adopted in [29] where the textual and visual features were first combined into a global vector following which the LSI technique was applied. In both these works, the two different kinds of image representations were simply combined together in a rather rigid way without any theoretical basis. Cai et al. [2] has used three image representations - viz., visual, textual, and link information, to construct a relationship graph of Web images. In this work, images were first clustered into different semantic groups by employing the textual and link features. This step was then followed by visual feature-based clustering of images in each semantic group. A problem in this two-step process is that an erroneous first step results

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

can propagate to the second step leading to poor image clustering. The use of visual, textual, and link features was also employed in [21]. Here, an iterative algorithm was applied to combine the co-clustering between images and surrounding text and the one-sided clustering of images based on visual features. The convergence property of this algorithm was not proven and the kind of combination is unsymmetrical according to the status of visual and textual features. From the above discussion, it is clear that the earlier efforts were along the direction of combining the information from visual and textual features instead of integrating them together synchronously under a sound theoretical framework.

In this paper, we propose the Consistent Isoperimetric High-Order Co-clustering (CIHC) framework for simultaneous integration of visual and textual features for efficient Web image clustering under a graph theoretical approach. Specifically, visual features, images and textual features are modelled as the three types of vertices of a tripartite graph. This tripartite graph is treated as two bipartite graphs of visual features & images and that of images & textual features from the surrounding texts of the image. Co-clustering is then achieved by simultaneously partitioning these two bipartite graphs together such that the information from both the kinds of features is optimally utilized. Note that the simultaneous partitioning of the two bipartite graphs is performed in such a way that the local clustering of each graph need not be optimal under the constraint that the fusion of the two results yields optimum image clustering. Actually, a similar concept was presented by Gao et al. [11] where the Consistent Bipartite Graph Co-partitioning (CBGC) was proposed under the spectral graph partitioning paradigm. An iterative algorithm using semi-definite programming (SDP) [1] is used to partition the tripartite graph which is computationally expensive and does not work well on large data sets. On the other hand, the proposed methodology requires a simple solution to a sparse system of overdetermined linear equations. Moreover, the CIHC framework has been derived from the isoperimetric graph partitioning approach which has been shown to achieve superior results than the spectral approach in terms of the quality, efficiency and stability of the partition [15, 16, 26]. Experimental results performed on images extracted from real Websites demonstrate the advantage of CIHC over CBGC in clustering Web images.

2. RELATED WORK

In this Section, we introduce some essential background on graph theory and review related work in the literature.

2.1 Homogeneous Graphs for clustering

An undirected homogeneous graph G ={V, E} consists of a set of vertices $V = \{v_1, v_2, ..., v_{|V|}\}$ and a set of edges E=$\{e_{ij}|$ edge between v_i and v_j, $i, j <= |V|\}$, where $|V|$ is the number of vertices. In a weighted graph, each edge e_{ij} has a positive weight denoted by $w(e_{ij})$. The weight of the edge signifies the level of association between the vertices. An edge weight of zero denotes the absence of an edge between the two respective vertices. Given a vertex numbering and the edge weights between the vertices, graphs can be represented by matrices. We begin with definitions of a few graph terminologies that play an essential role in the paper. The adjacency matrix **J** of the graph is defined as,

$$J_{ij} = \begin{cases} w(e_{ij}), & \text{if } e_{ij} \text{ exists} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

The degree of a vertex v_i denoted by d_i is defined as,

$$d_i = \sum_{e_{ij}} w(e_{ij}), \qquad \forall e_{ij} \in E \quad (2)$$

The degree matrix **D** of the graph is a diagonal matrix having degree of vertices along the diagonal while a degree vector **d** of a graph is a vector consisting of degree of all the vertices. The Laplacian matrix **L** of a graph is a symmetric matrix with one row and column for each vertex such that,

$$L_{v_i,v_j} = \begin{cases} d_i, & \text{if } i = j \\ -w(e_{ij}), & \text{if } e_{ij} \text{ exists} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Suppose we bipartition set V into subsets V_1 and V_2, then the corresponding graph *cut* is defined as,

$$cut(V_1, V_2) = \sum_{i \in V_1, j \in V_2} J_{ij} \quad (4)$$

The above definition can be extended to k-partitioning of the graph. The cut in which case is defined as,

$$cut(V_1, V_2, ..., V_k) = \sum_{n<\theta} cut(V_n, V_\theta) \quad (5)$$

A graph partitioning algorithm assigns a set of values to each vertex in the graph. We will refer to a vector consisting of the values for each of the vertices as the *indicator vector* of the graph. The *cutting* of the graph is dividing the indicator vector based on the values associated with each vertex using a splitting value. If **u** denotes the indicator vector of the graph and s is the splitting value, then the vertices are partitioned into the set of i such that $u_i > s$ and the set such that $u_i \leq s$. Spectral graph theory [4] which is based on performing eigen decomposition on matrices of the graphs, has been one of the most popular and widely applied graph partitioning methods. In [27], Shi and Malik have applied the spectral method to image segmentation. The objective function used in this work is,

$$\min \frac{\mathbf{x}^T \mathbf{L}\mathbf{x}}{\mathbf{x}^T \mathbf{D}\mathbf{x}}, \text{subject to } \mathbf{x}^T \mathbf{D}\mathbf{e} = 0,\ \mathbf{x} \neq \mathbf{0} \quad (6)$$

where **e** is a unit vector and **x** is a column vector such that $x_i = c_1$ if $i \in V_1$ and $x_i = -c_2$ if $i \in V_2$, where c_1 and c_2 are constants derived from the degree matrix **D**. By relaxing x_i from discrete to continuous, it can be shown that the solution to Equation (6) is the eigenvector corresponding to the second smallest eigenvalue λ_2 of the generalized eigenvalue problem [4, 12],

$$\mathbf{L}\mathbf{x} = \lambda \mathbf{D}\ \mathbf{x} \quad (7)$$

Partitions are then obtained by running a clustering algorithm such as k-means [17] on the eigenvector **x** corresponding to λ_2.

2.2 Bipartite Graph Model for pair-wise co-clustering

An undirected bipartite graph $G = \{M, W, E\}$, has two sets of vertices, viz., M and W and a set of graph edges E. Let **B** be an $|W|$ by $|M|$ graph weight matrix. An entry B_{ij} in this matrix is the weight of an edge appearing between a vertex $w_i \in W$ and a vertex $m_j \in M$. There are no edges between vertices of the same group. Then, the adjacency matrix of the bipartite graph is expressed as,

$$\mathbf{J} = \begin{bmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{0} \end{bmatrix} \quad (8)$$

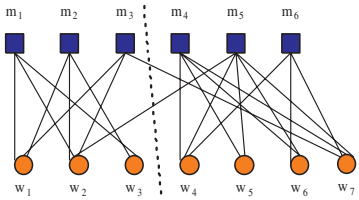

Figure 1: The square and circular vertices (m and w, respectively) denote the two data types in the co-clustering problem that are represented by the bipartite graph. Partitioning this bipartite graph leads to co-clustering of the two data types.

where the first $|W|$ vertices index W and the last $|M|$ index M. The two data types in the co-clustering problem can be represented by the two vertices of the weighted bipartite graph. Co-clustering of the data is achieved by partitioning the bipartite graph. In Figure 1, we show the bipartite graph partitioned using dotted lines. The two partitions obtained are $\{m_1, m_2, m_3, w_1, w_2, w_3\}$ and $\{m_4, m_5, m_6, w_4, w_5, w_6, w_7\}$, respectively. Therefore, the objects in M are clustered into $\{m_1, m_2, m_3\}$ and $\{m_4, m_5, m_6\}$, while those in W are clustered into $\{w_1, w_2, w_3\}$ and $\{w_4, w_5, w_6, w_7\}$ simultaneously. In order to compute these partitions using the spectral approach, we also need to solve a generalized eigenvalue problem as in Equation (7). However, due to the bipartite nature of the problem, the eigenvalue problem reduces to a much efficient Singular Value Decomposition (SVD) [12] problem, and has found application for co-clustering in varied fields [6, 19, 7]. Recently, Isoperimetric Co-clustering Algorithm (ICA) [26] was proposed to achieve pairwise co-clustering by partitioning a bipartite graph. ICA bears resemblance to the spectral approach in the sense that it does not require the coordinate information of the vertices of the graphs and allows us to find partitions of an optimal cardinality instead of a predefined cardinality. It has been shown that ICA outperforms the spectral approach in terms of the quality, efficiency and stability in partitioning a bipartite graph.

2.3 Tripartite Graph Model for triplet co-clustering

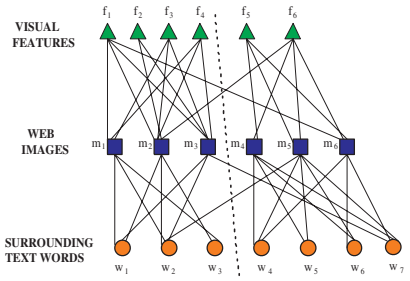

Figure 2: Tripartite graph of visual features, web images and the words coming from the surrounding text of the images.

The integration of multimodal information for efficient Web image clustering can be represented using a star-structured k-partite graph. The structure of this weighted graph is such that the central data type (images) is connected to all the other data types (different features). There are no direct edges between the feature vertices. Images are able to utilize multimodal information by being connected to the different kinds of feature vertices simultaneously. In an abstract level, a star-structured k-partite graph can be considered as a generalized version of a tripartite graph. As a preliminary attempt, in this work, we integrate visual and textual features simultaneously using a tripartite graph, shown in Figure 2. An undirected tripartite graph $G = \{F, M, W, E\}$, has three sets of vertices, viz., F, M and W with E as the set of edges. If $\mathbf{A}$ and $\mathbf{B}$ represent the weight matrices for feature-image and image-word bipartite graphs respectively, then the adjacency matrix of the graph is defined as,

$$\mathbf{J} = \begin{bmatrix} \mathbf{0} & \mathbf{A} & \mathbf{0} \\ \mathbf{A}^T & \mathbf{0} & \mathbf{B} \\ \mathbf{0} & \mathbf{B}^T & \mathbf{0} \end{bmatrix} \tag{9}$$

Every entry in $\mathbf{A}$ and $\mathbf{B}$ represents the importance of a particular feature and word for that image, respectively. For $\mathbf{B}$, word frequency in the surrounding text of the images is used. Partitioning this graph lets us achieve Web image clustering by integrating information from visual and textual features synchronously. Gao et al. [11] have used a similar tripartite graph model in their Consistent Bipartite Graph Co-partitioning (CBGC) framework. The tripartite graph is considered to be a fusion of the two bipartite graphs that are partitioned simultaneously using the spectral approach. Let $\mathbf{q} = [\mathbf{f} \quad \mathbf{m}]^T$ and $\mathbf{p} = [\mathbf{m} \quad \mathbf{w}]^T$ denote the indicator vectors for the two bipartite graphs and $\mathbf{D}^{(fm)}$, $\mathbf{D}^{(mw)}$, $\mathbf{L}^{(fm)}$ and $\mathbf{L}^{(mw)}$ represent the diagonal and Laplacian matrices of the two bipartite graphs. Then the objective function to minimize for the tripartite graph is expressed as a linear combination of objective functions of the two bipartite graphs as follows,

$$\min \left\{ \beta \frac{\mathbf{q}^T \mathbf{L}^{(fm)} \mathbf{q}}{\mathbf{q}^T \mathbf{D}^{(fm)} \mathbf{q}} + (1-\beta) \frac{\mathbf{p}^T \mathbf{L}^{(mw)} \mathbf{p}}{\mathbf{p}^T \mathbf{D}^{(mw)} \mathbf{p}} \right\} \tag{10}$$

$$\text{subject to } \mathbf{q}^T \mathbf{D}^{(fm)} \mathbf{e} = 0, \mathbf{q} \neq \mathbf{0}$$

$$\mathbf{p}^T \mathbf{D}^{(mw)} \mathbf{e} = 0, \mathbf{p} \neq \mathbf{0}$$

$$0 < \beta < 1$$

where the parameter β specifies the weightage for each bipartite graph in the linear combination. Illumined by this work and the recent results of ICA for pair-wise co-clustering [26], we propose to partition the visual feature-image-word tripartite graph using isoperimetric graph partitioning.

3. ISOPERIMETRIC GRAPH PARTITIONING FOR WEB IMAGE CLUSTERING

We partition the tripartite graph of features, web images and surrounding text words by extending the ICA framework. To proceed, we first provide a brief overview of ICA to partition a bipartite graph.

ICA has been motivated from the combinatorial formulation of the classic isoperimetric problem [8, 24, 15, 16]: *For a fixed area, find the shape with minimum perimeter.* It provides polynomial time heuristic for the NP-hard problem of finding a region with minimum perimeter for a fixed area. Let $V = \{M \bigcup W\}$ be the set of vertices of the bipartite graph. ICA partitions V into sets S and S^c, such that $S \bigcup S^c = V$ and $S \bigcap S^c = \phi$. Like other graph partitioning algorithms, ICA achieves optimum partitioning by finding S and S^c so that isoperimetric ratio of the graph h_G defined

as,

$$h_G = \frac{|\Delta S|}{Vol_S} \tag{11}$$

is minimized. The numerator and denominator represent the boundary area and the volume of S, respectively. The boundary of S is defined as,
$\Delta S = \{e_{ij} |$ edges between a vertex in S and $S^c\}$. Consequently,

$$|\triangle S| = \sum_{e_{ij} \in \triangle S} w(e_{ij}) \tag{12}$$

The combinatorial volume [8] can be defined as,

$$Vol_S = |S|, \tag{13}$$

After a few mathematical deductions, ICA achieves the minimization of Equation (11) by solving a sparse system of linear equations as,

$$\mathbf{L} \begin{bmatrix} \mathbf{m} \\ \mathbf{w} \end{bmatrix} = \mathbf{e} \tag{14}$$

where $\mathbf{L}$ is the Laplacian matrix of the bipartite graph, $[\mathbf{m}\ \mathbf{w}]^T$ is the indicator vector to indicate partitioning of the two types of vertices, and $\mathbf{e}$ is a vector of ones of size $|M| + |W|$. Solving this system of equations results in a real valued $[\mathbf{m}\ \mathbf{w}]^T$. In order to get partitions, this solution needs to be *cut* using a *splitting value* (as explained in Section 2).

To partition the visual feature-image-word tripartite graph, intuitively it might seem obvious to perform traditional extension of ICA (abbreviated as TICA) by solving a similar system of linear equations corresponding to the adjacency matrix defined in Equation (9) as follows,

$$\mathbf{L} \begin{bmatrix} \mathbf{f} \\ \mathbf{m} \\ \mathbf{w} \end{bmatrix} = \mathbf{e} \tag{15}$$

where $\mathbf{L}$, the indicator vector $[\mathbf{f}\ \mathbf{m}\ \mathbf{w}]^T$ and $\mathbf{e}$ are similarly defined for the tripartite graph. However, by doing so the visual feature-image-word tripartite graph actually ends up being a bipartite graph of image and visual feature & word. This can be seen by shifting the visual feature vertices on to the side of the word vertices as illustrated in Figure 3. Due to this, we will be unable to distinguish between cutting a visual feature-image edge and an image-word edge and is thus a conceptual misrepresentation of the basic structure of visual features, images and words as in Figure 2. An alternative approach is to use a weighting parameter α to prevent the mixing of visual feature and word vertices by defining the adjacency matrix of the tripartite graph as follows,

$$\mathbf{J} = \begin{bmatrix} \mathbf{0} & \mathbf{A} & \mathbf{0} \\ \mathbf{A}^T & \mathbf{0} & \alpha\mathbf{B} \\ \mathbf{0} & \alpha\mathbf{B}^T & \mathbf{0} \end{bmatrix} \tag{16}$$

However, as demonstrated in Section 3.1 and 5, the numerical weightage is unable to prevent the problem. Moreover, even if it works on a particular dataset, it is not possible to decide the numerical weight *a priori* and will not work across other datasets.

To overcome the ill-partitioning of Figure 3, we propose the Consistent Isoperimetric High-order Co-clustering (CIHC) to partition the visual feature-image-word tripartite graph by considering it as two bipartite graphs coupled together.

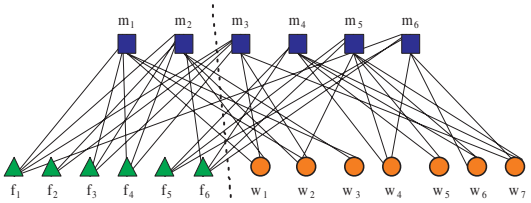

Figure 3: Traditional extension of ICA (TICA) for partitioning the feature-image-word tripartite graph ends up actually partitioning a bipartite graph of image and feature & word.

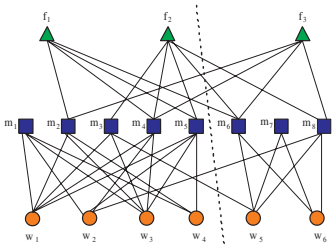

Figure 4: Toy problem of 3, 8 and 6 vertices of *F*, *M* and *W*, respectively with uniform weights along all edges. The dotted line shows the ideal cut for partitioning this graph.

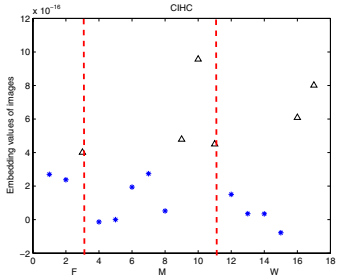

Figure 6: CIHC results for partitioning the toy graph. We are able to get perfect partitioning for each of *F*, *M* and *W* vertices.

It is easy to see that applying ICA separately on the two bipartite graphs will result in two different partitioning results on the images. In order to achieve consistent results, we need to partition the two bipartite graphs simultaneously. That is, visual feature-image bipartite graph needs to be partitioned under the constraints enforced on images by words while, the partitioning of image-words bipartite graph has to be under the constraints enforced on images by visual features. In other words, we achieve consistent partitioning of images under the constraints that the partitioning of visual feature-image or image-word need not be optimal. By doing so, we can consistently integrate the visual and textual features simultaneously for clustering the Web images.

Applying ICA to the visual feature-image bipartite graph, we get,

$$\mathbf{L}^{(fm)} \begin{bmatrix} \mathbf{f} \\ \mathbf{m} \end{bmatrix} = \mathbf{e}^{(fm)} \tag{17}$$

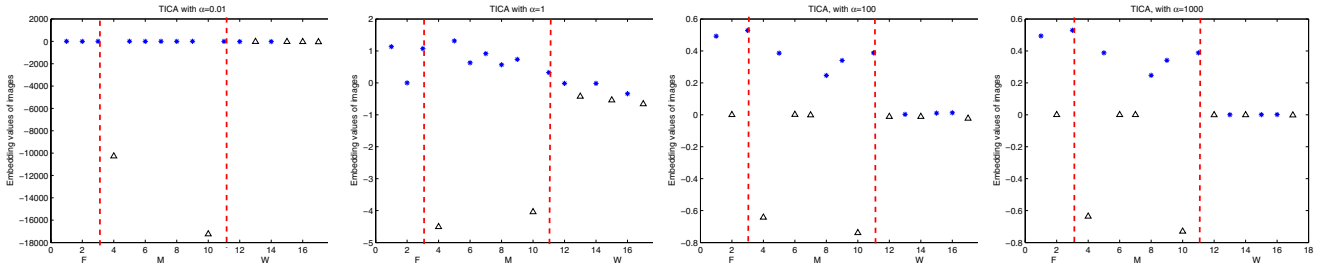

Figure 5: TICA results for partitioning the toy problem graph of Figure 4. Each sub-figure shows the results for $\alpha =$ 0.01, 1, 100 and 1000.

Similarly, image-word bipartite graph yields us,

$$\mathbf{L}^{(mw)} \begin{bmatrix} \mathbf{m} \\ \mathbf{w} \end{bmatrix} = \mathbf{e}^{(mw)} \tag{18}$$

We combine the above two system of linear equations as,

$$\begin{bmatrix} \mathbf{L}^{(fm)} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}^{(mw)} \end{bmatrix} \begin{bmatrix} \mathbf{f} \\ \mathbf{m} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{e}^{(fm)} \\ \mathbf{e}^{(mw)} \end{bmatrix} \tag{19}$$

$$\mathbf{F}\,\mathbf{r} = \mathbf{v}$$

where $\mathbf{v}$ is a vector of ones of size $|F| + 2|M| + |W|$. Note that, $\mathbf{F}$ is not a square matrix, i.e. this is an overdetermined system of linear equations where the number of equations is more than the number of variables. Overdetermined system of linear equations is usually inconsistent and does not have any solution. However, many least squares methods exist to approximate the solution [20]. We adopted the QR decomposition method due to its simplicity and efficiency to solve Equation (19). Notice that, any other method can be employed as well. Amongst the common methods for cutting the indicator vector are the median cut and the ratio cut. Median cut uses the median of the indicator vector $\mathbf{r}$ as the *splitting value* to produce equally sized partitions while ratio cut chooses one such that the resulting partitions have the lowest isoperimetric ratio of the graph indicating optimal partitioning. As our goal is not to necessarily produce equally sized clusters, we employ the ratio cut to get a bipartition.

3.1 Illustrative Toy problem

Above, we discussed the need for partitioning the tripartite graph by simultaneously partitioning the two bipartite graphs. We now illustrate this on a toy problem that TICA does not yield optimum results. For this, we created a tripartite graph shown in Figure 4 having 3 F, 8 M and 6 W vertices with uniform weights along all the edges. It is easy to infer the ideal partitioning of this graph shown in the Figure using a dotted line. We partitioned this graph using TICA with adjacency matrix defined in Equation (16). In Figure 5, we show the corresponding results achieved when we varied the value of α from 0.01 to 1000. The X-axis has the vertices in the order of F, M and W with the dotted line separating each of them. Embedding value for each vertex in the indicator vector is plotted along the Y-axis. The two pattern plots ($*$ and $\triangle$) represent the two clusters obtained. These results clearly show that in spite of increasing edge weights for one of the bipartite graphs drastically over the other, TICA is still unable to distinguish between cutting an F-M edge from an M-W edge on a simple graph. On the other hand, Figure 6 shows the results achieved using CIHC. We can see that, CIHC achieves perfect clustering for all the vertices. Through this toy problem, we have shown the need to consistently apply isoperimetric co-clustering simultaneously. More results, on real Web images will be presented in Section 5.

3.2 Algorithm Summary

The main steps of CIHC can be summarized as follows:

1. Using weight matrix $\mathbf{A}$ of the visual feature-image bipartite graph, construct the Laplacian matrix $\mathbf{L}^{(fm)}$.
2. Similarly, using weight matrix $\mathbf{B}$ of the image-word bipartite graph, construct $\mathbf{L}^{(mw)}$.
3. Using Equation (19), construct $\mathbf{F}$, $\mathbf{r}$ and $\mathbf{v}$, and solve the system of linear equations $\mathbf{F}\,\mathbf{r} = \mathbf{v}$.
4. Employ ratio cut on $\mathbf{r}$ to get the partitions.

3.3 Advantages of CIHC over CBGC

Both CIHC and CBGC partition the visual feature-image-word tripartite graph by considering it as a fusion of the two bipartite graphs. However, as we explain below the CIHC framework has a number of advantages over CBGC. In CBGC, the spectral objective functions of the two bipartite graphs are transformed into single-objective function of Equation (10) by expressing it as a weighted linear combination. As argued in [9], this is a very ad-hoc approach and not a principled one. The two objective functions represent two different kinds of information. Hence, instead of converting the original dual-objective problem into a single objective problem, one should use a dual-objective algorithm directly. Another drawback of CBGC, is that its performance is dependent on three parameters, viz. β, θ_1 and θ_2. β is the weighting parameter used in the linear combination of objective functions while θ_1 and θ_2 are parameters used to put constraints on the SDP bound controllers. The problem here is that these three parameters have to be predetermined. In their work, the authors test the performance of CBGC on few image categories by varying the values of the parameters and choose the best values. As we show in our results (Section 5), this approach for tuning the CBGC parameters works on only those few categories and performs poorly on the other datasets. In the real world application for Web image clustering, it is impossible to know what parameter values are to be chosen for clustering the images retrieved. On the other hand, CIHC is a completely parameter-less

approach and does not require *a priori* specification of any parameters. In terms of computational complexity, CIHC is much more efficient compared to CBGC as it only requires a simple solution to a sparse system of linear equations.

4. THEORETICAL ANALYSIS OF CIHC

4.1 Time Complexity

Computational time required by CIHC depends on the solution to Equation (19). In particular, the time complexity is dependent on the number of non-zero entries in $\mathbf{L}^{(fm)}$ and $\mathbf{L}^{(mw)}$, which asymptotically is $O(|E|)$ where E is the set of edges in the tripartite graph. Note that, this only measures the time complexity to compute the indicator vector. We also need to include the time complexity to employ the ratio cut which is of the order of $O(h\, logh)$ where $h = |F| + |M| + |W|$. Factoring this in, the time complexity of CIHC is $O(|E|+h\, logh)$. Empirical results on computational speed are presented in Section 5.3.

4.2 Sensitivity Analysis

In this section, we analyze the sensitivity of CIHC and CBGC with respect to a general parameter ρ.

4.2.1 CIHC

Recall from Equation (19), CIHC requires a solution to a sparse system of linear equations represented by,

$$\mathbf{F}\ \mathbf{r} = \mathbf{v} \tag{20}$$

Differentiating this equation with respect to ρ,

$$\mathbf{F}\frac{\delta \mathbf{r}}{\delta \rho} = -\mathbf{r}\frac{\delta \mathbf{F}}{\delta \rho} + \frac{\delta \mathbf{v}}{\delta \rho} \tag{21}$$

For a given solution to Equation (20), $\mathbf{F}$ and $\mathbf{r}$ are known and $\frac{\delta \mathbf{F}}{\delta \rho}$ can be determined analytically. In order to determine the derivative at point $\mathbf{r}$, $\frac{\delta \mathbf{r}}{\delta \rho}$ can be solved for as a system of linear equations.

4.2.2 CBGC

The CBGC objective function is,

$$\min\left\{\beta\frac{\mathbf{q}^T\mathbf{L}^{(fm)}\mathbf{q}}{\mathbf{q}^T\mathbf{D}^{(fm)}\mathbf{q}} + (1-\beta)\frac{\mathbf{p}^T\mathbf{L}^{(mw)}\mathbf{p}}{\mathbf{p}^T\mathbf{D}^{(mw)}\mathbf{p}}\right\} \quad \text{subject to certain constraints} \tag{22}$$

If we drop the constant weighting parameter β and consider only the first term, we get,

$$\min\left\{\frac{\mathbf{q}^T\mathbf{L}^{(fm)}\mathbf{q}}{\mathbf{q}^T\mathbf{D}^{(fm)}\mathbf{q}}\right\} \tag{23}$$

With reference to the previous discussion on Spectral graph partitioning in Section 2, we know from Equations (6) and (7) that the solution for minimizing this term is the eigenvector corresponding to second smallest eigenvalue λ_2 of the generalized eigenvalue problem,

$$\mathbf{L}^{(fm)}\ \mathbf{q} = \lambda_2\ \mathbf{D}^{(fm)}\ \mathbf{q} \tag{24}$$

Differentiating this equation with respect to ρ,

$$\mathbf{q}\frac{\delta \mathbf{L}^{(fm)}}{\delta \rho} + \mathbf{L}^{(fm)}\frac{\delta \mathbf{q}}{\delta \rho} = \mathbf{D}^{(fm)}\ \mathbf{q}\frac{\delta \lambda_2}{\delta \rho} + \lambda_2\mathbf{q}\frac{\delta \mathbf{D}^{(fm)}}{\delta \rho} + \lambda_2\mathbf{D}^{(fm)}\frac{\delta \mathbf{q}}{\delta \rho} \tag{25}$$

Using Rayleigh quotient and the Chain rule, it is possible to calculate $\frac{\delta \lambda_2}{\delta \rho}$.

The Rayleigh quotient is,

$$\lambda = \frac{\mathbf{q}^T\mathbf{L}^{(fm)}\mathbf{q}}{\mathbf{q}^T\mathbf{q}} \tag{26}$$

Applying Chain rule,

$$\frac{\delta \lambda_2}{\delta \rho} = \frac{\delta \lambda_2}{\delta \mathbf{q}}\frac{\delta \mathbf{q}}{\delta \rho} \tag{27}$$

In the above equation, $\frac{\delta \lambda_2}{\delta \mathbf{q}}$ can be calculated from equation (26) as,

$$\frac{\delta \lambda_2}{\delta \mathbf{q}} = 2\mathbf{L}^{(fm)}\mathbf{q}(\mathbf{q}^T\mathbf{q})^{-1} - 2\mathbf{q}^T\mathbf{L}^{(fm)}\mathbf{q}(\mathbf{q}^T\mathbf{q})^{-2}\mathbf{q} \tag{28}$$

From Equation (25), since all the terms are either known or can be calculated analytically, we get a system of linear equations which may be solved for $\frac{\delta \mathbf{q}}{\delta \rho}$.

If we now consider the second term in Equation (22), and proceed similarly, we get

$$\mathbf{p}\frac{\delta \mathbf{L}^{(mw)}}{\delta \rho} + \mathbf{L}^{(mw)}\frac{\delta \mathbf{p}}{\delta \rho} = \mathbf{D}^{(mw)}\ \mathbf{p}\frac{\delta \lambda_2}{\delta \rho} + \lambda_2\mathbf{p}\frac{\delta \mathbf{D}^{(mw)}}{\delta \rho} + \lambda_2\mathbf{D}^{(mw)}\frac{\delta \mathbf{p}}{\delta \rho} \tag{29}$$

We can analyze the effect of a specific parameter, e.g., edge weight, by substituting for the general parameter ρ. Equations (21), (25) and (29) show that the derivative of the CIHC solution is never degenerate. On the other hand, the CBGC solution may be degenerate depending on the value of λ_2 and the state of its corresponding eigenvector.

5. EXPERIMENTS AND RESULTS

5.1 Data Preparation

For dataset preparation we followed the same approach as in [11]. A web crawler was first sent out to crawl some of the Webpages in the Yahoo directory[1] to extract images and their surrounding texts. Images having width-height ratios larger than 5 or less than 1/5 and images having both width and height less than 60 pixels were removed. After this, the image database consisted of 15,000 images which were manually assigned to 42 categories.

The surrounding text extracted was filtered to remove common stop words such as conjunctions, articles, etc. The words left over after this were considered to be the textual features of the image. We observed that the quality of the surrounding text varies depending on the source Webpage from where an image was extracted. Correspondingly, we refer to the textual features for each category to be "poor", "average" or "good". For the experiments, we randomly selected 10 of these categories shown in Table I. To ensure that the image-text bipartite graph is not disconnected when two

[1]http://dir.yahoo.com/Arts/Visual_Arts/Photography/

Table I: Image categories used in the experiments

Category Name	Category Size	Text Quality
Owls	71	Average
Flowers	64	Average
Lions	56	Average
Elephants	85	Good
Horses	76	Average
Snow Mountains	82	Average
Flying Eagle	67	Average
Dusk	58	Poor
Plants	79	Poor
Railways	77	Good

categories are mixed together, we added an extra dummy word to the graph that connects to all the images. The weights for each of these edges was set to be the reciprocal of the number of images in the graph. For the visual features, we used the PCA-SIFT[2] image descriptors [18].

5.2 Web Image clustering results

We first demonstrate the need to integrate both the visual and textual features simultaneously. We present the results for clustering Web images using only textual features and only visual features and then show that the performance can be improved by integrating the two simultaneously using CIHC framework. We will also compare image clustering results of CIHC with TICA and CBGC.

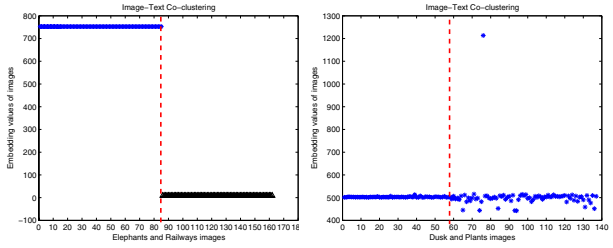

Figure 7: Image-Text Co-clustering. (Left) Both *Elephants* and *Railways* have "good" textual features leading to perfect clustering. (Right) *Dusk* and *Plants* have "poor" textual features that are not sufficient in separating the two categories. Most of the Web images have noisy surrounding text and is not by itself sufficient for clustering.

For Web image clustering using only surrounding text, we selected 4 image categories viz., *Elephants*, *Trains*, *Dusk* and *Plants*. We mixed *Elephants* & *Trains* that both have "good" textual features and *Dusk* & *Plants* that have "poor". We applied the ICA algorithm to the Image-Text bipartite graph to get the partitions. For all the experiments, we have used the volume as per Equation (13). In Figure 7, we show the embedding values of the images and the partitioning obtained. The dotted line separates the image categories. As expected in the case of *Elephants* & *Trains*, we get perfect clustering results. However, *Dusk* & *Plants* images were extracted from some photo gallery webpages that hardly had any worthwhile text in them. Images from these two categories had a lot of noisy text such as *photographer name*, *photographer address*, *copyrights*, etc. Consequently, the clustering result on these categories using only surrounding text is very poor. This experiment shows that if the textual features of the image categories are very good, then those themselves are enough to yield optimum results. However, that is not always true with the surrounding text around the Web images.

[2]http://www.cs.cmu.edu/~yke/pcasift/

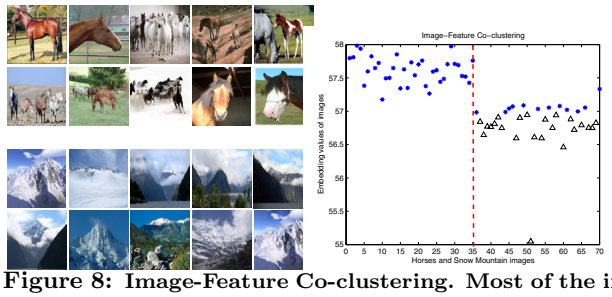

Figure 8: Image-Feature Co-clustering. Most of the images selected from *Horses* and *Snow Mountains* are separable using only the visual features. Images from the *Horses* category that were misclassified were due to their similar backgrounds such as sky or snow.

With regards to Web image clustering by relying only on visual features, we selected 4 categories viz., *Horses*, *SnowMountains*, *Owls* and *FlyingEagle*. For this experiment, we first mixed *Horses* and *SnowMountains* by selecting 35 visually dissimilar images from each of them. In Figure 8, we show sample images from both these categories and the image clustering results achieved by applying ICA to the Image-Feature bipartite graph. Images from these two categories are mainly dissimilar with few similarities in some images such as the sky or greenery in the background. Due to this fact, we are able to separate the two categories to a great extent using only the visual features. On the other hand, although *Owls* and *FlyingEagle* belong to two different semantic categories, their images resemble on a number of grounds as shown in Figure 9. Images from these categories have a bird-like object with similar backgrounds. Owing to the semantic gap, the visual features are unable to represent the semantics accurately. Figures 7, 8 and 9 show that integrating the low-level visual feature representation and high-level semantics coming from the textual features together simultaneously may yield better clustering results.

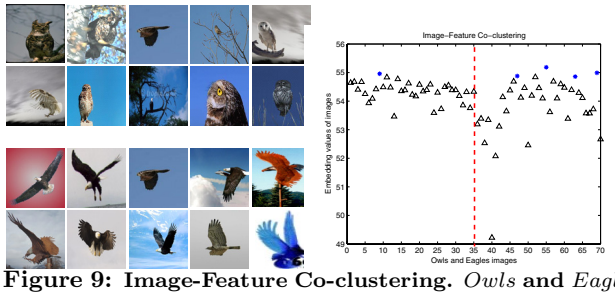

Figure 9: Image-Feature Co-clustering. *Owls* and *Eagles* are inseparable using only the visual features due to the semantic gap. Images from both the categories have similar backgrounds and a bird-like structure at the center of the image.

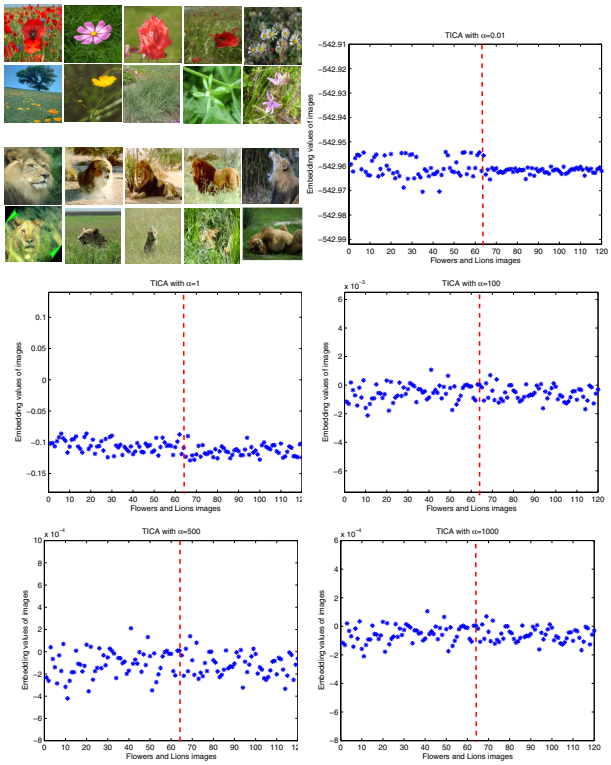

Figure 10: TICA is unable to separate *Flowers* and *Lions* images despite the varying values of α from 0.01 to 1000. Images from both these categories have "average" textual features and visually dissimilar images. TICA performs disastrously due to the ill-partitioning of the tripartite graph explained in Figure 3

In Section 3, we had discussed the need for simultaneously partitioning the two bipartite graphs of visual features & images and images & textual features. We now show experimentally that TICA does not yield optimum results for clustering Web images. For this, we mixed *Flowers* and *Lions* images. Sample images from these two categories and the results obtained using TICA are shown in Figure 10. Both these categories have "average" textual features with images having similar backgrounds. The value of the weighting parameter α was varied from 0.01 to $1,000$. This experiment demonstrates that TICA actually ends up partitioning the bipartite graph of images and the two features together. As was the case with the toy problem results (Section 3.1), even the change in the values of α does not help in getting better results. Figure 11 displays the results we get for partitioning the same dataset using CIHC. From these results, we can make two key observations. First of all, we can see the advantages of utilizing the information from both kinds of features simultaneously. In most of the real-world scenarios, it is unlikely that either of the features would be capable of completely describing the respective categories on their own. "Average" textual features with tolerable amount of noise along with visual features is good enough to get decent clustering results. Secondly, the capability of CIHC to achieve the same can be seen.

We have compared CIHC with CBGC in terms of the image clustering results, scalability and computational speed.

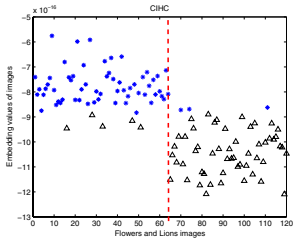

Figure 11: CIHC performs well in grouping *Flowers* and *Lions* images by successfully integrating the visual and textual features simultaneously.

As mentioned in Section 3.3, CBGC is a very parameter-dependent framework which relies heavily on the values of θ_1, θ_2 and β. To decide on the values for these 3 parameters, we followed the approach suggested by the authors [11]. We randomly selected two image categories (*Elephants* and *SnowMoutains*) and varied the values of all the parameters between 0 and 1 for the partitioning. Values that gave best clustering result were chosen and used for the rest of the categories. The problem with CBGC is that these values are category specific and have to be retuned for the other categories. As an example, in Figure 12, we show the clustering results for *Elephants* and *SnowMountains* using CBGC and CIHC. Based on the above parameter values, CBGC is able to get perfect clustering results. On the same dataset, CIHC gets similar results. In another case involving *Flying Eagle* and *Lions* shown in Figure 13, CBGC has misclassified a number of *Lions* images. CIHC on the other hand was able to generate very good clusters. The same is true in the case of *Flowers* and *Horses* shown in Figure 14 where CBGC produces disastrous results and is completely outperformed by CIHC. We observed similar results in the clustering of other categories. To summarize, in CBGC, parameters tuned for partitioning certain categories produce excellent results for those particular categories and may not work well for others. CIHC framework instead does not require any parameter values to be pre-determined.

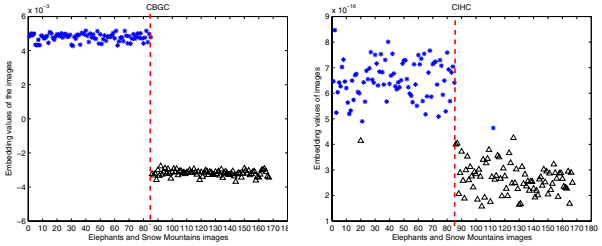

Figure 12: In the clustering of *Elephants* and *Snow Moutains*, CBGC is able to get perfect clustering due to the fact that the values used for the parameters β, θ_1 and θ_2 are suitable for separating these two categories. CIHC is devoid of any parameters and is able to get comparable results on the same dataset.

To evaluate the image clustering performance of CIHC and CBGC across all the categories, we have used the cross-accuracy metric [11]. If two image categories having n_1 and n_2 images respectively are mixed, then the ground truth

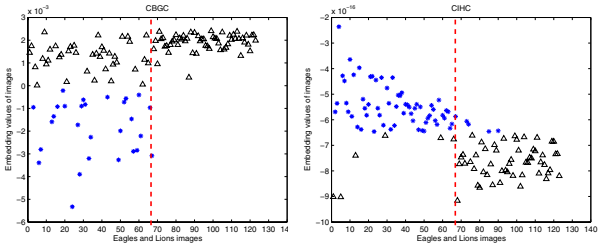

Figure 13: Parameters previously set for CBGC are unable to separate *Flying Eagles* and *Lions*. A number of *Lions* images are misclassified. On the other hand, CIHC performs well.

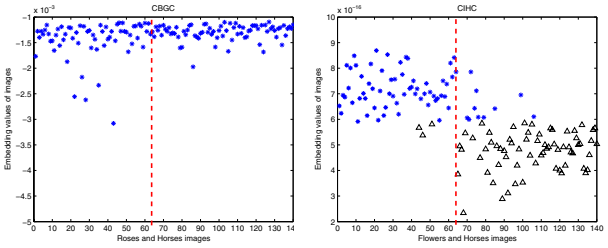

Figure 14: Another example of the effect of a set of parameter values on other image categories. The set parameter values of CBGC perform very poorly in separating *Flowers* and *Horses*. Once again, as can be seen CIHC gets decent results.

Boolean vector **rt** can be written as,

$$\mathbf{rt} = (1, 1, ..., 1, 0, 0, ..., 0) \tag{30}$$

where the first n_1 elements are set to 1 and the rest n_2 elements are set to 0. The image clustering results can be represented as a Boolean vector **rc** having the same ordering of elements as **rt**. Cross-accuracy is defined as follows,

$$accuracy = \max\left\{\frac{\sum_i (rt_i \oplus rc_i)}{n_1 + n_2}, 1 - \frac{\sum_i (rt_i \oplus rc_i)}{n_1 + n_2}\right\} \tag{31}$$

where $\oplus$ represents the exclusive-OR operation. We mixed every image category with the rest of the category and measured the accuracy of the clustering. In Figure 15, we have plotted the accuracy of CIHC (Y-axis) vs CBGC (X-axis). Each circle in the plot represents a possible image category pair. It can be seen that most of the circles fall in the upper part of the diagonal. This indicates that in the clustering of most of the image category pairs, CIHC outperforms CBGC. The few circles on the lower part of the diagonal are the category pairs for which CBGC has been properly tuned with the parameter values. In Table II, we show the mean accuracy between each category and all other categories for both the algorithms. CIHC has a higher mean accuracy and outperforms CBGC on all the categories.

5.3 Computational Speed

We now compare the computational speed of CIHC with CBGC. The time take by both algorithms is dependent on the sparseness of the two data matrices. In other words, it takes more time to partition a densely connected tripartite graph compared to a sparsely connected one. For this reason, we considered the worst case scenario of a fully connected tripartite graph (with uniform weights) where every vertex in both the bipartite graphs is connected with all other vertices of the other type. Since the time required to cut the indicator vector is the same for both algorithms, we compare on the basis of the time required to calculate the indicator vector. The algorithms were implemented using MATLAB 7.0[3]. For CBGC implementation, we made use of the SDP library SDPA-M [4] [10]. The experiment was performed on a machine with a 3 GHz Intel Pentium 4 processor with 1 GB RAM. In Figure 16, we plot the time required by the algorithms as the number of vertices in the fully connected tripartite graph increases. Time for CIHC gradually

Table II: Average clustering performance

Category Name	CBGC	CIHC
Owls	0.5898	0.8241
Flowers	0.6248	0.8342
Lions	0.6544	0.8132
Elephants	0.8706	0.8941
Horses	0.6363	0.8244
Snow Mountains	0.6050	0.7746
Flying Eagle	0.6728	0.8608
Dusk	0.5412	0.5614
Plants	0.6386	0.6543
Railways	0.6631	0.8070

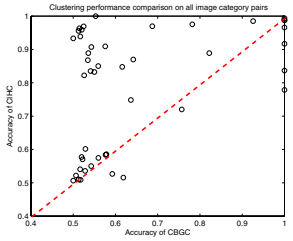

Figure 15: Clustering performance of CBGC and CIHC on all image category pairs. Each circle represents a possible category pair. Most of the circles fall in the upper part of the diagonal.

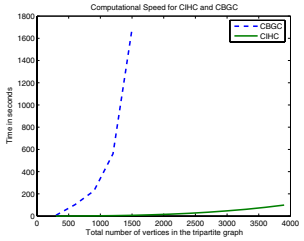

Figure 16: Computational speed comparison of CIHC with CBGC. The time required by each of the algorithms to compute the indicator vector are displayed for increasing number of vertices in the tripartite graph.

[3]http://www.mathworks.com
[4]http://grid.r.dendai.ac.jp/sdpa/

increases with the number of vertices. For the maximum number of vertices we increased to - about almost $4,000$, CIHC required about 98 seconds only. CBGC on the other hand, was unable to keep up with CIHC. As can be seen, the time required by CBGC really shoots up for a few hundred vertices in the graph. Moreover, CBGC is unscalable and is unable to handle larger sized graphs, which was also verified by [23, 22]. In our experiment, CBGC was unable to handle graphs with more than $1,500$ vertices. This experiment clearly demonstrates the computational efficiency of CIHC and the potential for applicability in large-scale real-world applications.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we addressed the problem of Web image clustering by simultaneous integration of visual and textual features from a graph partitioning perspective. In particular, we modelled visual features, images, and words from the surrounding text of the images using a tripartite graph. This graph is actually considered as a fusion of two bipartite graphs that are partitioned simultaneously by the proposed CIHC framework. Although a similar approach has been adopted before, the main contribution of this work lies in the computational efficiency, quality in Web image clustering and scalability to large image repositories that CIHC is able to achieve. We demonstrate this through experimental results performed on real Web images.

In future work, there are a number of directions we are actively pursuing. Currently, in order to get more than two partitions, we recursively apply CIHC which is a common approach in many other graph partitioning algorithms [11, 26, 15, 28]. We are currently investigating methods to get more than two clusters directly. Another extension of this work is to get flexible clusterings for each of the vertex types in the tripartite graph. That is, currently we have a hard partitioning framework where there is a one-one association between features, images and words belonging to one cluster. It would be interesting to discover the association between visual features grouped in one cluster and words or images grouped in another. We are also working on having different number of partitions for each of visual features, images and words, instead of all having the same. To this end, we have found the recent work on matrix factorization by Long et al. [23, 22] very interesting and helpful.

7. ACKNOWLEDGMENTS

This research was partially funded by the 21^{st} Century Jobs Fund Award, State of Michigan, under grant: 06-1-P1-0193, and by National Science Foundation, under grant: IIS-0713315.

8. REFERENCES

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

[2] D. Cai, X. He, Z. Li, W.-Y. Ma, and J.-R. Wen. Hierarchical clustering of www image search results using visual, textual and link information. In *proc. of ACM Multimedia*, 2004.

[3] M. Cascia, S. Sethi, and S. Sclaroff. Combining textual and visual cues for content-based image retrieval on the world wide web. In *proc. of IEEE CBAIVL*, 1998.

[4] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[5] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[6] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *proc. KDD*, 2001.

[7] C. H. Q. Ding. Unsupervised feature selection via two-way ordering in gene expression analysis. *Bioinformatics*, 19:1259 – 1266, 2003.

[8] J. Dodziuk. Difference equations, isoperimetric inequality and the transience of certain random walks. *Transactions of the American Mathematical Society*, 284:787–794, 1984.

[9] A. A. Freitas. A critical review of multi-objective optimization in data mining: a position paper. *SIGKDD Explor. Newsl.*, 6(2):77–86, 2004.

[10] K. Fujisawa, Y. Futakata, M. Kojima, S. Matsuyama, S. Nakamura, K. Nakata, and M. Yamashita. Sdpa-m (semidefinite programming algorithm in matlab) user's manual - version 6.2.0. Technical report, Dept. Math. & Comp. Sciences, Tokyo Institute of Technology, http://grid.r.dendai.ac.jp/sdpa/publication.html, 2000.

[11] B. Gao, T.-Y. Liu, T. Qin, X. Zheng, Q.-S. Cheng, and W.-Y. Ma. Web image clustering by consistent utilization of visual features and surrounding texts. In *proc. of ACM Multimedia*, 2005.

[12] G. H. Golub and C. F. Van-Loan. *Matrix Computations*. John Hopkins Press, 1989.

[13] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice Hall, Upper Saddle River, NJ, 2002.

[14] S. Gordon, H. Greenspan, and J. Goldberger. Applying the information bottleneck principle to unsupervised clustering of discrete and continuous image representations. In *proc. of IEEE ICCV*, 2003.

[15] L. Grady and E. L. Schwartz. Isoperimetric graph partitioning for image segmentation. *IEEE Transactions on PAMI*, 28(3):469– 475, 2006.

[16] L. Grady and E. L. Schwartz. Isoperimetric partitioning: A new algorithm for graph partitioning. *SIAM Journal on Scientific Computing*, 27(6):1844–1866, 2006.

[17] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[18] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *proc. of IEEE CVPR*, 2004.

[19] R. Kumar, U. Mahadevan, and D. Sivakumar. A graph-theoretic approach to extract storylines from search results. In *proc. of ACM KDD*, 2004.

[20] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Soc for Industrial and Applied Math, 1995.

[21] Z. Li, G. Xu, M. Li, W.-Y. Ma, and H.-J. Zhang. Grouping www image search results by novel inhomogeneous clustering method. In *proc. of MMM*, 2005.

[22] B. Long, X. Wu, Z. Zhang, and P. S. Yu. Unsupervised learning on k-partite graphs. In *proc. of ACM KDD*, 2006.

[23] B. Long, Z. Zhang, X. Wu, and P. S. Yu. Spectral clustering for multi-type relational data. In *proc. of ICML*, 2006.

[24] B. Mohar. Isoperimetric numbers of graphs. *Journal of Combinatorial Theory, Series B*, 47:274–291, 1989.

[25] G. Qiu. Image and feature co-clustering. In *proc. of IEEE ICPR*, 2004.

[26] M. Rege, M. Dong, and F. Fotouhi. Co-clustering documents and words using bipartite isoperimetric graph partitioning. In *proc. of IEEE ICDM*, 2006.

[27] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans PAMI*, 22(8):888 – 905, 2000.

[28] H. Zha, X. He, C. H. Q. Ding, H. Simon, and M. Gu. Bipartite graph partitioning and data clustering. In *proc. of ACM CIKM*, 2001.

[29] R. Zhao and W. I. Grosky. Narrowing the semantic gap - improved text-based web document retrieval using visual features. *IEEE Trans. on Multimedia*, 4(2):189–200, 2002.