A Min-max Cut Algorithm for Graph Partitioning and Data Clustering

Chris H.Q. Ding[a], Xiaofeng He[a,b], Hongyuan Zha[b], Ming Gu[c], Horst D. Simon[a]
[a] NERSC Division, Lawrence Berkeley National Laboratory
University of California, Berkeley, CA 94720
[b] Department of Computer Science and Engineering
Pennsylvania State University, University Park, PA 16802
[c] Department of Mathematics
University of California, Berkeley, CA 94720
{chqding,hdsimon}@lbl.gov, {xhe,zha}@cse.psu.edu, mgu@math.berkeley.edu

Abstract

An important application of graph partitioning is data clustering using a graph model — the pairwise similarities between all data objects form a weighted graph adjacency matrix that contains all necessary information for clustering. Here we propose a new algorithm for graph partition with an objective function that follows the min-max clustering principle. The relaxed version of the optimization of the min-max cut objective function leads to the Fiedler vector in spectral graph partition. Theoretical analyses of min-max cut indicate that it leads to balanced partitions, and lower bonds are derived. The min-max cut algorithm is tested on newsgroup datasets and is found to outperform other current popular partitioning/clustering methods. The linkage-based refinements in the algorithm further improve the quality of clustering substantially. We also demonstrate that the linearized search order based on linkage differential is better than that based on the Fiedler vector, providing another effective partition method.

1 Introduction

Graph partitioning has very broad range of applications. At one end are the near-regular graphs, the mesh of a 2D surface of an airfoil or a 3D engine cylinder. Partitioning such a mesh into subdomains for distributed memory processors is a common task. Popular software packages for this partitioning task are developed [17, 16]. At another end are the graphs generated from the World Wide Web. These graphs are highly irregular or random, and node degrees vary dramatically. Partitioning the web graph is useful to automatically identify topics from the retrieved webpages for a user query [15].

Here we emphasize graph partition as data clustering using a graph model. Given the attributes (coordinates) of the data points in a dataset and the similarity or affinity metric between any two points, the symmetric matrix containing similarities between all pairs of points forms a weighted adjacency matrix (weight matrix) of an undirected graph. Thus the data clustering problem becomes a graph partition problem.

The data clustering point of view of graph partitioning helps to define more appropriate criteria for partitioning. In the simplest MINcut algorithm, a connected graph is partitioned into two subgraphs with the cutsize (cut set) minimized. However, MINcut often results in a skewed cut, i.e., a very small subgraph is cut away [4]. Various constraints are introduced, such as the *ratio cut* [4, 14], the *normalized cut* [22], etc. to circumvent the problem. However, skewed cuts still occur when the overlaps between clusters are large.

In this paper, we propose a new graph partition method based on the min-max clustering principle: the similarity or association between two subgraphs (cut set) is minimized, while the similarity or association within each subgraph (summation of similarity between all pairs of nodes within a subgraph) is maximized. These two requirements can be satisfied simultaneously with a simple min-max cut function. We present a number of theoretical analyses of min-max cut, and show that min-max cut always leads to more balanced cuts than the ratio cut and the normalized cut.

Like many other methods, the optimal solution to the graph partition problem is NP-complete because of the combinatoric nature of the problem. An effective approach is to consider continuous relaxation of such problems. An example is to compute a principal direction/component (principal eigenvector of the weight matrix), and find a cut point along this direction so that all points on one side belong to one subgraph, and all points on the other side belong to another subgraph. This establishes a linear search order on which the min-max cut can be efficiently applied to search the optimal cut.

The relaxed version of the min-max cut function optimization leads to a generalized eigenvalue problem. The second lowest eigenvector, also called the Fiedler vector, provides a linear search order (Fiedler order). Thus the min-max cut algorithm (we call it Mcut algorithm) provides both a well-defined objective and a clear procedure to search for the optimal solution. We test the algorithm on a number of newsgroup text datasets and compare it with several current methods. The Mcut algorithm outperforms others.

We introduce a linkage difference metric that effec-

0-7695-1119-8/01 $17.00 © 2001 IEEE

tively identifies nodes near the cut. We find many nodes sitting on the wrong side of the optimal cutpoint, i.e., they have higher linkage to the other cluster than the one they are currently assigned to. Swapping them to the correct side, the objective function is reduced and the clustering accuracy is improved substantially.

It is generally believed that the Fiedler order provides the best known linearized order to search for the optimal cut. Here we find a linkage differential order that provides a *better* ordination than the Fiedler order. Searching based on linkage differential order consistently outperforms those based on the Fiedler order. The linkage differential order start from any existing clustering results and iteratively improves the ordering and therefore the clustering.

2 Min-max cut

Given a weighted graph $G = G(E, V)$ with node set V, edge set E and weight matrix W, we wish to partition it into two subgraphs A, B using the min-max clustering principle — minimize similarity between clusters and maximize similarity within a cluster. This is a sound principle well established in statistics, data mining and machine learning areas. The similarity or association between two nodes is their edge weight W_{uv}. Thus the similarity between subgraphs A, B is the cutsize

$$\mathrm{cut}(A, B) = W(A, B) \qquad (1)$$

where

$$W(A, B) = \sum_{u \in A, v \in B} W_{uv}, \quad W(A) \equiv W(A, A). \qquad (2)$$

Similarity or association within a cluster (subgraph A) is the sum of all edge weights within A: $W(A)$. Note that the weight W_{uu} on node u is included in $W(A)$, which is important for some applications. Thus the min-max clustering principle requires we minimize $\mathrm{cut}(A, B)$ while maximizing $W(A)$ and $W(B)$ at the same time. All these requirements can be simultaneously satisfied by the following objective function,

$$\mathrm{Mcut} = \frac{\mathrm{cut}(A, B)}{W(A)} + \frac{\mathrm{cut}(A, B)}{W(B)}. \qquad (3)$$

We call this new objective function the min-max cut function or Mcut for short. Mcut is inspired by previous works on spectral graph partition [21, 14, 22] (see section 3). It turns out that the continuous relaxation of Eq.(3) must be solved in a way that is different from existing graph partition relaxations [21, 14, 22]. To reveal the solution, we reorder the rows and columns of W conformally with subgraphs A and B such that

$$W = \begin{pmatrix} W_A & W_{A,B} \\ W_{B,A} & W_B \end{pmatrix}. \qquad (4)$$

Let $\mathbf{x}$ and $\mathbf{y}$ be vectors conformally partitioned with A and B, i.e., $\mathbf{x} = (1 \cdots 1, 0 \cdots 0)^T$, $\mathbf{y} = (0 \cdots 0, 1 \cdots 1)^T$. It follows from Eqs.(1,2) that

$$\begin{aligned} \mathrm{cut}(A, B) &= \mathbf{x}^T (D - W) \mathbf{x} = \mathbf{y}^T (D - W) \mathbf{y}, \quad (5) \\ W(A) &= \mathbf{x}^T W \mathbf{x}, \quad W(B) = \mathbf{y}^T W \mathbf{y}. \end{aligned}$$

Hence the objective function (3) can be rewritten as

$$\mathrm{Mcut} = \frac{\mathbf{x}^T (D - W) \mathbf{x}}{\mathbf{x}^T W \mathbf{x}} + \frac{\mathbf{y}^T (D - W) \mathbf{y}}{\mathbf{y}^T W \mathbf{y}}. \qquad (6)$$

Observe that in Eq.(6), Mcut is invariant under changes of $||\mathbf{x}||_2$ and $||\mathbf{y}||_2$, and

$$\mathbf{x}^T D \mathbf{y} = 0 \quad \text{and} \quad \mathbf{x}^T W \mathbf{x} > 0, \mathbf{y}^T W \mathbf{y} > 0.$$

Taking these relations into account, we obtain a useful lower bound on (3) in Theorem 1 below. Observe that the problem Eq.(6) can be relaxed into the following optimization problem

$$\min \quad \frac{\widehat{\mathbf{x}}^T (I - \widehat{W}) \widehat{\mathbf{x}}}{\widehat{\mathbf{x}}^T \widehat{W} \widehat{\mathbf{x}}} + \frac{\widehat{\mathbf{y}}^T (I - \widehat{W}) \widehat{\mathbf{y}}}{\widehat{\mathbf{y}}^T \widehat{W} \widehat{\mathbf{y}}} \qquad (7)$$

subject to $||\widehat{\mathbf{x}}||_2 = ||\widehat{\mathbf{y}}||_2 = 1$, $\widehat{\mathbf{x}}^T \widehat{\mathbf{y}} = 0$, $\widehat{\mathbf{x}}^T \widehat{W} \widehat{\mathbf{x}} > 0$, $\widehat{\mathbf{y}}^T \widehat{W} \widehat{\mathbf{y}} > 0$, where $\widehat{W} = D^{-1/2} W D^{-1/2}$ and $\widehat{\mathbf{x}} = D^{1/2}\mathbf{x}/|D|^{1/2}$, $\widehat{\mathbf{y}} = D^{1/2}\mathbf{y}/|D|^{1/2}$. The conditions that $\widehat{\mathbf{x}}^T \widehat{W} \widehat{\mathbf{x}} > 0$ and $\widehat{\mathbf{y}}^T \widehat{W} \widehat{\mathbf{y}} > 0$ are necessary since $\widehat{W}$ in general is an indefinite matrix. Let the largest 2 eigenvalues of $\widehat{W}$ be λ_1, λ_2. $\lambda_1 = 1$ by construction. We have the following (proof omitted).

Theorem 1. Assume that $\lambda_1 + \lambda_2 > 0$. Let vectors $\widehat{\mathbf{x}}$ and $\widehat{\mathbf{y}}$ solve problem Eq.(7). Choose $\widehat{U}$ to be any column orthogonal matrix such that $\widehat{Q} = \left(\widehat{\mathbf{x}}, \widehat{\mathbf{y}}, \widehat{U}\right)$ is an $n \times n$ orthogonal matrix. Then $\widehat{Q}^T \widehat{W} \widehat{Q} = \begin{pmatrix} \begin{pmatrix} \alpha & \gamma \\ \gamma & \alpha \end{pmatrix} & \mathbf{0} \\ \mathbf{0} & \widetilde{W} \end{pmatrix}$, where $\alpha = (\lambda_1 + \lambda_2)/2$, $|\gamma| = |\lambda_1 - \lambda_2|/2$.

It follows from Theorem 1 that both ratios of (7) are equal at the optimal solution:

$$\frac{\widehat{\mathbf{x}}^T (I - \widehat{W}) \widehat{\mathbf{x}}}{\widehat{\mathbf{x}}^T \widehat{W} \widehat{\mathbf{x}}} = \frac{\widehat{\mathbf{y}}^T (I - \widehat{W}) \widehat{\mathbf{y}}}{\widehat{\mathbf{y}}^T \widehat{W} \widehat{\mathbf{y}}} = \frac{2}{\lambda_1 + \lambda_2} - 1, \qquad (8)$$

and the optimal value is $\mathrm{Mcut} = 4/(\lambda_1 + \lambda_2) - 2$.

Since Eq.(7) is a continuous relaxation of Eq.(3), the fact that the two terms in (7) are equal at optimal solution suggests that the two terms of Eq.(3) should also be rather "close" to each other, implying $W(A)$ should be "close" to $W(B)$. Hence the resulting clusters tend to have similar weights and are thus balanced. This is one indication that Mcut is a desired objective function for data clustering (see sections 4 and 7 for more discussions).

2.1 Fiedler linear search order

The solution to partition problem can be represented by an indicator vector $\mathbf{q}$, where the nodal value of $\mathbf{q}$ on node u is $q_u = \{a, -b\}$, depending on $u \in A$ or B. Finding the optimal partition is NP-complete. A well-known and effective solution is to first compute a linear search order and then find a cut point along this index order that minimizes the Mcut objective. Theorem 1 implies that the solution vectors $\mathbf{x}, \mathbf{y}$ must lie in the eigenspace of $\widehat{W}$. The first eigenvector $\mathbf{z}_1 = D^{1/2} e$, $e = (1, \cdots, 1)^T$ with the largest eigenvalue $\lambda_1 = 1$ does not match $\mathbf{q}$. The second eigenvector $\mathbf{z}_2$ of $\widehat{W}$ satisfies $\mathbf{z}_2^T \mathbf{z}_1 = 0$ and has positive and negative elements, therefore is a good approximation of $\mathbf{q}$.

More directly, we can show that

$$\min_{\mathbf{q}} \text{Mcut}(A, B) = \min_{\mathbf{q}} \frac{J_N(A, B)}{1 - J_N(A, B)/2} \Rightarrow \min_{\mathbf{q}} J_N(A, B)$$

where

$$J_N(A, B) \equiv J_N(\mathbf{q}) = \frac{\mathbf{q}^T (D - W)\mathbf{q}}{\mathbf{q}^T D \mathbf{q}}. \qquad (9)$$

Relaxing q_u to real number in $[-1, 1]$, the solution for minimizing Rayleigh quotient $J_N(\mathbf{q})$ is given by

$$(D - W)\mathbf{q} = \zeta D \mathbf{q}, \qquad (10)$$

subject to $\mathbf{q}^T \mathbf{e} = 0$. The solution to this generalized eigenvalue problem is the second eigenvector $\mathbf{q}_2$, called the Fiedler vector, and the corresponing eigenvalue ζ_2 is called the Fiedler value. Sorting the Fiedler vector provides the desired linear search order. Furthermore, we obtain a lower bound for the Mcut objective,

$$\min_{\mathbf{q}} \text{Mcut}(A, B) \geq \frac{\zeta_2}{1 - \zeta_2/2}. \qquad (11)$$

Note this bound is the same as the optimal Mcut value in Eq.(8), because $\zeta_i = 1 - \lambda_i$, and $4/(\lambda_1 + \lambda_2) - 2 = 4/(2 - \zeta_2) - 2 = \zeta_2/(1 - \zeta_2/2)$.

3 Related work on spectral graph partition

Spectral graph partitioning is based on the properties of eigenvectors of the Laplacian matrix $L = D - W$, first developed by Donath and Hoffman [8] and Fiedler [11, 12], and recently populated by the work of Pothen, Simon and Liu [21]. The objective of the partitioning is to minimize the cut size $J(A, B) = \text{cut}(A, B)$ with the requirement that two subgraphs have the same number of nodes: $|A| = |B|$. Using indicator variable x_u, $x_u = \{1, -1\}$ depending on $u \in A$ or B, the cutsize is

$$\text{cut}(A, B) = \sum_{e_{uv} \in E} \frac{(x_u - x_v)^2}{4} W_{uv} = \frac{\mathbf{x}^T (D - W)\mathbf{x}}{2}. \qquad (12)$$

Relax x_u from $\{1, -1\}$ to continuous value in $[-1, 1]$, miniming cut(A, B) is equivelent to solve the eigensystem

$$(D - W)\mathbf{x} = \lambda \mathbf{x}. \qquad (13)$$

Since the trivial $\mathbf{x}_1 = \mathbf{e}$ is associated with $\lambda_1 = 0$, the second eigenvector $\mathbf{x}_2$, the Fiedler vector, is the solution.

Hagen and Kahng [14] remove the requirement $|A| = |B|$ and show that the Fiedler vector provides a good linear search oerder to the ratio cut (Rcut) partitioning criteria[4]

$$\text{Rcut} = \frac{\text{cut}(A, B)}{|A|} + \frac{\text{cut}(A, B)}{|B|}. \qquad (14)$$

The use of generalized eigensystem from Eq.(13) to Eq.(10) has been studied by a number of authors [9, 5, 22]. Chung [5] especially emphasizes the advantage of using normalized Laplacian matrix which leads to Eq.(10). Shi and Malik [22] propose the normalized cut,

$$\text{Ncut} = \frac{\text{cut}(A, B)}{\deg(A)} + \frac{\text{cut}(A, B)}{\deg(B)} \qquad (15)$$

where $\deg(A) = \sum_{u \in A} d_u$ is the sum of node degrees, which is also called the *volumn* [5] of subgraph A, in contrast to the size of A. They show that Ncut can be reduced to $\text{Ncut}(A, B) = J_N(\mathbf{q})$ in Eq.(9). Therefore, Ncut uses the same linear search order based on $\mathbf{q}_2$ as Mcut objective. Following the same analysis, we obtain a lower bound for the Ncut objective,

$$\min_{\mathbf{q}} \text{Ncut}(A, B) \geq \zeta_2. \qquad (16)$$

Here Rcut, Ncut and Mcut objective functions are first *prescribed* by motivating considerations and then the linear order of the Fiedler vector of (normalized) Laplacian matrix is argued to be the appropriate search order (by relaxing discrete indicator variables). It is important to note that the same objective functions can be automatically obtained as the eigenvalues of the Fiedler vector using a perturbation analysis on the (normalized) Laplacian matrix [6]. This further strengthens the connection between objective function and the Fiedler vector.

Beside spectral partitioning methods, other recent partitioning methods seek to minimize the sum of subgraph diameters, see [7] or k-center problem [1] for examples. There are other clustering methods that use singular value decompositions, for example [10].

4 Random graph model

Perhaps the most important feature of the Mcut method is that it tends to produce balanced cut, i.e., the resulting clusters (subgraphs) have similar sizes. Here we use the random graph model [3, 4] to illustrate this point. Suppose we have a uniformly distributed random graph with n nodes. For this random graph, any two nodes are connected with probability p. We consider the four objective functions, the MINcut, Rcut, Ncut and Mcut. We have the following

Theorem 2. For random graphs, MINcut favors highly skewed cuts, i.e., very uneven sizes. Mcut favors balanced cut, i.e., both subgraphs have the same sizes. Rcut and Ncut show no size preferences.

Proof. We compute the object functions for the partition of G into A and B. Note that the number of edges between A and B are $p|A||B|$ on average. We have

$$\text{MINcut}(A, B) = p|A||B|$$

For Rcut, we have

$$\text{Rcut}(A, B) = \frac{p|A||B|}{|A|} + \frac{p|A||B|}{|B|} = p(|A| + |B|) = np.$$

For Ncut, since all nodes have the same degree $(n-1)p$,

$$\text{Ncut}(A, B) = \frac{p|A||B|}{p|A|(n-1)} + \frac{p|A||B|}{p|B|(n-1)} = n/(n-1).$$

For Mcut, we have

$$\text{Mcut}(A, B) \;=\; \frac{|B|}{|A|-1} + \frac{|A|}{|B|-1}$$

We now minimize these objectives. Clearly, MINcut favors $|A| = n-1$ and $|B| = 1$ or $|B| = n-1$ and $|A| = 1$, both are skewed cuts. Minimizing Mcut(A, B), we obtain a balanced cut: $|A| = |B|$. Rcut and Ncut objectives have no size dependency and no size preference, which also implies possible unstable results. This completes the proof.

5 Mcut algorithm

The algorithm for partitioning a graph into two subgraphs becomes the following.

1. Compute the Fiedler vector from Eq.(10). Sort nodal values to obtain the Fiedler order.
2. Search for the optimal cut point corresponding to the lowest Mcut based on the Fiedler order.
3. Do linkage-based refinements (see section 8).

The computation of the Fiedler vector can be quickly done via the Lanczos method [20]. A fast software package for this calculation, LANSO, is available online (http://www.nersc.gov/~kewu/planso.html). The Lanczos iteration has computational complexity of $O(|E| + |V|)$.

6 Experiments

Document clustering has been popular in analyzing text information. Here we perform experiments on newsgroup articles in 20 newsgroups. We focus on three datasets, each has two newsgroups:

```
1/2:    alt.atheism/comp.graphics
10/11:  rec.sport.baseball/rec.sport.hockey
18/19:  talk.politics.mideast/talk.politics.misc
```

(The newsgroup dataset together with the `bow` toolkit for processing is available online[19]).

Word-document matrix X is first constructed. 2000 words are selected according to the mutual information between words and documents

$$I(w) = \sum_d p(w, d) \log_2 [p(w, d)/p(w)p(d)]$$

where w represents a word and d represents a document. Words are stemmed using [19]. Standard `tf.idf` scheme for term weighting is used and standard cosine similarity between two documents d_1, d_2 $\text{sim}(d_1, d_2) = d_1 \cdot d_2/|d_1||d_2|$ is used. When each document, column of X, is normalized to 1 using L_2 norm, document-document similarities are calculated as $W = X^T X$. W is interpreted as the weight/affinity matrix of the undirected graph. From this similarity matrix, we perform the clustering as explained above.

For comparison purpose, we also consider three other clustering methods: the ratio cut [4, 14], the normalized cut [22] (see section 3) and the principle direction divisive partitioning (PDDP) [2]. PDDP is based on the idea of principle component analysis (PCA) applied to the vector-space model on X. First X is centered, i.e., the average of each row (a word) is subtracted. Then the first principle component is computed. The loadings of the documents (the projection of each document on the principle axis) form a 1-dim linear search order. This provides a heuristic very similar to the linear search order provided by the Fiedler vector. Instead of searching through to find a minimum based on some objective function, PDDP partitions data into two parts at the center of mass.

To increase statistics, we perform these two-cluster experiments in a way similar to cross-validation. We divide one newsgroup A randomly into K_1 subgroups and the other newsgroup B randomly into K_2 subgroups. Then one of the K_1 subgroups of A is mixed with one of the K_2 subgroups of B to produce a dataset G. The graph partition methods are run on this dataset G to produce two clusters. Since the true label of each newsgroup article is known, we use accuracy, percentage of newsgroup articles correctly clustered, as a measure of success. This is repeated for all $K_1 K_2$ pairs between A and B, and the accuracy is averaged. In this way, every newsgroup articles is used the same number of times. The mean and standard deviation of accuracy are listed.

In Table 1, the clustering results are listed for balanced cases, i.e., both subgroups have about 200 newsgroup articles. Mcut performs about the same as Ncut for newsgroups NG1/NG2, where the cluster overlap is small. Mcut performs substantially better than Ncut for newsgroups NG10/NG11 and newsgroups NG18/NG19, where the cluster overlaps are large. Mcut performs slightly better than PDDP. Rcut always performs the worst among the 4 methods and will not be studied further.

In Table 2, the clustering results are listed for unbalanced cases, i.e., one subgroup has about 300 newsgroup articles and another subgroup has about 200. This is generally a harder problem due to the unbalanced prior distributions. In this case, both Mcut and Ncut perform reasonably well, no clear deterioration is seen, while the performance of PDDP clearly deteriorated. This indicates the strength of Mcut method using graph model. Mcut consistently performs better than Ncut for cases where the cluster overlaps are large.

Dataset	Mcut	Ncut	Rcut	PDDP
NG1/NG2	97.2±1.1	97.2±0.8	63.2±16.2	96.4±1.2
NG10/NG11	79.5±11.0	74.4±20.4	54.9±2.5	89.1±4.7
NG18/NG19	83.6±2.5	57.5±0.9	53.6±3.1	71.9±5.4

Table 1: Accuracy (%) of clustering experiments using Mcut, Rcut, Ncut and PDDP. Each test set G is a mixture of 400 news articles, 200 from each newsgroup.

Dataset	Mcut	Ncut	PDDP
NG1/NG2	97.6 ± 0.8 %	97.2± 0.8 %	90.6 ± 2.1%
NG10/NG11	85.7 ± 8.3 %	73.8± 16.6 %	87.4 ± 2.6%
NG18/NG19	78.8 ± 4.5 %	65.7± 0.5 %	59.6 ± 2.4%

Table 2: Accuracy of clustering experiments using Mcut, Ncut and PDDP. Each test set G is a mixture of 300 news articles from one newsgroup and 200 news articles from the other newsgroup.

7 Skewed cut

We further study the reasons that Mcut consistently outperforms Ncut in large overlap cases. A key observation is that Ncut can be written as

$$\text{Ncut} = \frac{\text{cut}(A,B)}{W(A)+\text{cut}(A,B)} + \frac{\text{cut}(A,B)}{W(B)+\text{cut}(A,B)}, \quad (17)$$

since $\deg(A) \equiv \sum_{u \in A,\ v \in G} W_{uv} = W(A) + \text{cut}(A,B)$. Thus, Ncut sometimes cuts out a set with a very small weight, i.e., a skewed cut, because $\text{cut}(A,B)$ in the denominators help to produce a smaller Ncut value.

We examine several cases and one specific case is shown in Figure 1. The cut points for Mcut and Ncut and relevant quantity are listed in Table 3. Ncut has two pronounced valleys, and produces a skewed cut. while Mcut has a very flat valley and gives balanced cuts. Further examination shows that in both cases, the cutsizes obtained in Ncut are equal or bigger than the weight (self-similarity) of the smaller cluster as listed in Table 3. In the example of Figure 1, Ncut produces a cutsize of 262.7, much larger than the weight $W(B,B) = 169$. In these cases, clearly the Ncut objective [see Eq.(17)] is not appropriate. In the Mcut objective, the cutsize is absent in the denominators; this provides a balanced cut.

These case studies provide some insights to these graph partition methods. Prompted by these studies, here we provide further analysis and derive general conditions under which a skewed cut will occur. Consider the balanced cases where $W(A) \simeq W(B)$. Let

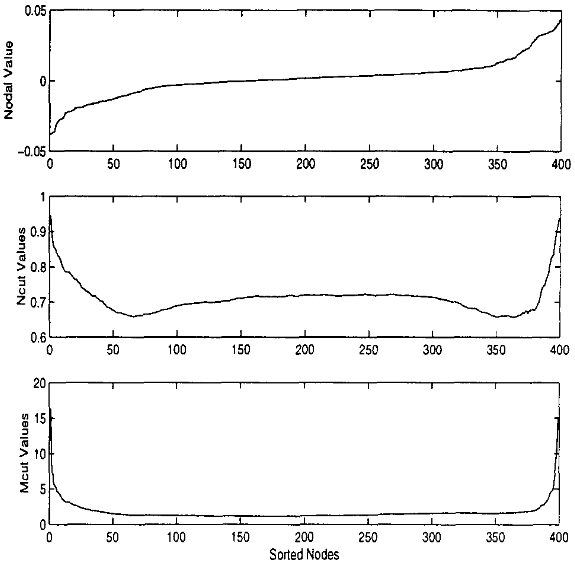

Figure 1: Top: Nodal values of sorted Fiedler vector. Middle: Ncut values as the cut point moves from $i_{cut} = 1, 2, ...n$. Bottom: Mcut values. A dataset from NG18/NG19 in Table 1.

Method	i_{cut}	$\text{cut}(A,B)$	W(A,A)	W(B,B)
Ncut	364	262.7	5312.6	169.0
Mcut	141	1026.6	1488.9	2464.7

Table 3: Cut point, cutsize, within cluster similarities for the case in Figure 1.

$$\text{cut}(A,B) = f \cdot \langle W \rangle, \quad \langle W \rangle = \frac{1}{2}(W(A) + W(B)),$$

where $f > 0$ is the average fraction of cut relative to within cluster associations.

In the case when the partition is optimal, A and B are exactly the partitioning result. The corresponding Ncut value is

$$\text{Ncut}_0 = \frac{\text{cut}(A,B)}{W(A)+\text{cut}(A,B)} + \frac{\text{cut}(A,B)}{W(B)+\text{cut}(A,B)} \simeq \frac{2f}{1+f} \quad (18)$$

For a skewed partition A_1, B_1, we have $W(A_1) \ll W(B_1)$, and therefore $\text{cut}(A_1, B_1) \ll W(B_1)$. The corresponding Ncut value is

$$\text{Ncut}_1 \simeq \frac{\text{cut}(A_1,B_1)}{W(A_1)+\text{cut}(A_1,B_1)}. \quad (19)$$

Using Ncut, a skewed or incorrect cut will happen if $\text{Ncut}_1 < \text{Ncut}_0$. Using Eqs.(18, 19), this condition is satisfied if

$$\text{Ncut}: \quad W(A_1) \geq (\frac{1}{2f} - \frac{1}{2})\text{cut}(A_1, B_1)$$

We can repeat the same analysis using Mcut and calculating Mcut_0 and Mcut_1. The condition for a skewed cut using Mcut is $\mathrm{Mcut}_1 < \mathrm{Mcut}_0$, which is

$$\mathrm{Mcut}: \quad W(A_1) \geq \frac{1}{2f} \operatorname{cut}(A_1, B_1).$$

For large overlap case, say, $f = 1/2$, the conditions for possible skewed cut are:

$$\begin{aligned} \mathrm{Ncut}: \quad W(A_1) &\geq \operatorname{cut}(A_1, B_1)/2 \\ \mathrm{Mcut}: \quad W(A_1) &\geq \operatorname{cut}(A_1, B_1) \end{aligned}$$

The relevant quantity is listed in Table 4. For datasets newsgroups 10-11, and newsgroups 18-19, the condition for skewed Ncut is satisfied most of the time, leading to many skewed cuts and therefore lower clustering accuracy in Tables 1,2. For the same datasets, condition for skewed Mcut is not satisfied most of time, leading to more correct cuts and therefore higher clustering accuracy.

Dataset	cut(A,B)	W(A,A)	W(B,B)	f
NG1/NG2	549.4	1766.4	1412.5	0.346
NG10/NG11	772.8	1372.8	1581.0	0.523
NG18/NG19	1049.5	2093.9	1665.5	0.558

Table 4: Average values of cut(A,B), W(A,A), W(B,B) and the fraction in three datasets using Mcut.

8 Linkage-based refinements

The heuristic linear search order provided by the Fiedler vector is generally a good heuristic, as the results shown above. Nevertheless, it may not necessarily be the perfect one. Here we explore this point and find an effective refinement method which substantially improves the quality of graph partitioning.

The linear search order provided by sorting the Fiedler vector $\mathbf{q}$ implies that nodes on one side of the cut point must belong to one cluster: if $q_u \geq q_v \geq q_w$ where u, v, w are nodes, then the linear search will not allow the situation that u, w belong to one cluster and v belongs to the other cluster. Such a strict order is not necessary. In fact, in large overlap cases, we expect some nodes could be moved to the other side of the cut while lowering the overall objective function.

How to identify those nodes near the cut? For this purpose, we define linkage ℓ as a closeness or similarity measure between two subgraphs (clusters):

$$\ell(A, B) = W(A, B)/W(A)W(B) \qquad (20)$$

here $W(A), W(B)$ are for normalization purpose so that $\ell(A, B)$ is insensitive to cluster weights (this is motivated by the *average linkage* $\ell(A, B) = W(A, B)/|A||B|$ in hierarchical agglomerative clustering. Following the spirit of min-max cut, we replaced $|A|, |B|$ by $W(A)$, $W(B)$). For a single node u, its linkage to subgraph A is $\ell(A, u) = W(A, u)/W(A)$. Now we can identify the nodes near the cut. If a node u is well inside a cluster, u will have a large linkage with the cluster, and a small linkage with the other cluster. If u is near the cut, its linkages with both clusters should be close. Therefore, we define the linkage difference

$$\Delta\ell(u) = \ell(u, A) - \ell(u, B). \qquad (21)$$

A node with small $\Delta\ell$ should be near the cut and is a possible candidate to be moved to the other cluster.

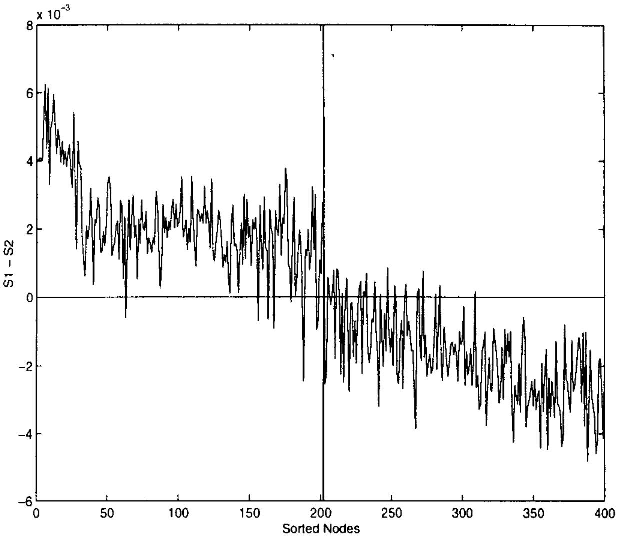

Figure 2: Linkage difference of all nodes. The vertical line indicates the cutpoint using Mcut. Nodes on the left forms cluster A and nodes on the right forms cluster B.

In Figure 2, we show linkage difference $\Delta\ell$ for all nodes. The vertical line is the cut point. It is interesting to observe that not only many nodes have small $\Delta\ell$, but quite a number of nodes whose $\Delta\ell$ have the wrong signs (e.g., $\Delta\ell(u) < 0$ if $u \in A$, or, $\Delta\ell(v) > 0$ if $v \in B$). For example, node #62 has a relatively large negative $\Delta\ell$. This implies node #62 has a larger linkage to cluster B even though it is located in cluster A (left of the cutpoint). Indeed, if we move node #62 to cluster B, the objective function is reduced. Therefore we find a better solution.

After moving node #62 to cluster B, we try to move another node with negative $\Delta\ell$ from cluster A to cluster B depending on whether the objective function is lowered. In fact, we move all nodes in cluster A with negative $\Delta\ell$ to cluster B if the objective function is lowered. Similarly we move all nodes in cluster B with positive $\Delta\ell$ to cluster A. This procedure of swapping nodes is called the "linkage-based swap". It is implemented by sorting the array $s(u)\Delta\ell(u)$ [$s(u) = -1$ if $u \in A$ and $s(u) = 1$ if $u \in B$] in decreasing order to provide a priority list and then moving the nodes, one by one. The greedy move starts from the top of the list to the last node u where $s(u)\Delta\ell(u) \geq 0$. This swap reduces the objective function and increases the partitioning quality. In Table 5, the effects on clustering accuracy due to the swap are listed. In all cases, the accuracy increases. Note that in the large overlap cases, NG9/NG10, NG18/NG19, the accuracy increase about 10% over the Mcut without refinement.

If $s(u)\Delta\ell(u) < 0$ but close to 0, node u is in the correct cluster, although it is close to the cut. Thus we select the smallest 5% of the nodes with $s(u)\Delta\ell(u) < 0$ as the candidates, and move those which reduce Mcut objective to the other cluster. This is done in both cluster A and B. We call this procedure "linkage-based move". Again, these moves reduce Mcut objective and therefore improve the solution. In Table 5, their effects on improving clustering accuracy are shown. Adding together, the linkage bassed refinements improve the accuracy by 20%. Note the final Mcut results are about 30-50% better than Ncut and about 6-25% better than PDDP (see Tables 5 and 1).

Dataset	Mcut	+Swap	+Swap+Move
NG1/NG2	97.2 ± 1.1%	97.5 ± 0.8 %	97.8 ±0.7%
NG10/NG11	79.5 ±11.0%	85.0 ± 8.9 %	94.1 ±2.2%
NG18/NG19	83.6 ± 2.5%	87.8 ± 2.0 %	90.0 ±1.4%

Table 5: Improvements of clustering accuracy due to linkage-based refinements for Mcut alone, Mcut plus swap, and Mcut plus swap and move over 5% smallest $\Delta\ell$ on both sides of the cutpoint.

9 Linkage differential order

It is generally believed that the Fiedler order provides the best known linearized order to search for the optimal cut (although delicate counter examples exist [13, 23]). Is there a linear search order better than the Fiedler order?

Our analysis in previous sections suggests a new linear search order. Given the linkage difference in Figure 2, we see that quite a few nodes far away from the cut point have wrong $\Delta\ell$ signs, that is, they belong to the other subgraph. This strongly suggests that the Fiedler order is not necessarily the best linear search order. In fact, we can sort linkage difference $\Delta\ell$ to obtain a linear order different from the Fiedler order, which will be referred to as *linkage differential* (LD) order. The search to find the best Mcut cut point based on this new LD order represents another improvement over the standard Mcut method.

The results are given in Table 6. We see that the Mcut values obtained on this new order are lower than that based on the Fiedler order. The clustering accuracy also increases substantially. The quality of the clustering based on this new order is slightly better than the results obtained by using Mcut+swap in Table 5. Therefore, we find a new linear order that leads to better graph partitioning than that provided by the Fiedler order.

Note that the LD order does not depend on the Fiedler order. For example, we can obtain the LD order based on the principal direction in PDDP. Furthermore, the LD order can be recursively applied to the clustering results obtained from an earlier LD order for further improvements.

Dataset	Acc(F)	Acc(LD)	Min(F)	Min(LD)
NG1/NG2	97.2 ± 1.1%	97.6±0.8%	0.698	0.694
NG10/NG11	79.5 ±11.0%	87.2±8.0%	1.186	1.087
NG18/NG19	83.6 ± 2.5%	89.2±1.8%	1.126	1.057

Table 6: Improvements on accuracy (2nd and 3rd columns) due to the linkage differential (LD) order over Fiedler order (F). Improvements on min(Mcut) values are also shown. (4th and 5th columns).

10 Hierarchical divisive Mcut

So far in this paper, we focus on bisection a graph into two subgraphs. If more subgraphs or clusters are desired, one can recursively apply Mcut and related refinement to each subgraph, until certain stopping criteria is met, either the desired number of clusters is reached or min(Mcut) value is above certain pre-defined value.

Once the recursive division is stopped, some refinements along the lines discussed in section 8 should be applied. This is because even if during each bisection step, all nodes are optimally partitioned, the final partition is not necessarily optimal, since they are not obtained directly according to the optimal Mcut objective

$$\text{Mcut}_K = \frac{\text{cut}(G_1, \bar{G}_1)}{W(G_1)} + \frac{\text{cut}(G_2, \bar{G}_2)}{W(G_2)} + \cdots + \frac{\text{cut}(G_K, \bar{G}_K)}{W(G_K)}.$$

when G is partitioned into K subgraphs, $G_1, \cdots, G_K$. Note that for $K \geq 3$, each term in Mcut_K will be larger than that in $K = 2$ cases because $\text{cut}(G_p, \bar{G}_p), p = 1, \cdots K$ will increase on average while the weight (self-similarity) $W(G_p)$ will decrease. Thus, Mcut_K would differ from Ncut_K [22] much more than in the $K = 2$ cases [cf. Eq.(17)]. From the analysis regarding balanced cuts in previous sections, Ncut is more likely to produce skewed cuts. Therefore, Mcut is essential in $K-$way partition.

When applying the refinements on $K \geq 3$ clusters, one may apply the 2-way linkage-based refinement *pairwisely* on all pairs of clusters[18]. However, a direct K-way linkage-based refinement procedure may be adopted: Assume a node u currently belongs to cluster G_i. The linkage difference $\Delta\ell_{ij}(u) = \ell(u, G_j) - \ell(u, G_i)$ for all other $K-1$ clusters are computed. The smallest $\Delta\ell_{ij}(u)$ and the corresponding cluster id are stored as an entry in a priority list. This is repeated for all nodes so every entry of the list is filled. The list is then sorted according to $\Delta\ell_{ij}(u)$ to obtain the final priority list. Following the list, nodes are then moved one after another to the appropriate clusters if the overall Mcut_K objective is reduced. This completes one pass. For $K \geq 3$, several passes may be necessary.

11 Summary

We introduce the Mcut algorithm for graph partition. It is shown that the min-max objective function follows the clustering principle and produces balanced partitions, compared to many skewed cuts produced by Ratio cut, Nornmalized cut and PDDP algorithms in cases of large cluster overlaps. The linkage difference metric effectively identifies those nodes near the cut, which leads to effective refinement procedures. Finally, the new linkage differential order is shown to provide a better linear search order than the best known Fiedler order. Many datasets such as text information are represented by bipartite graphs. Mcut algorithm can also be applied to the bi-clustering model[24] on these bipartite graph problems.

Acknowledgements. This work is supported in part by Office of Science, Office of Laboratory Policy and Infrastructure, of Department of Energy under contract DE-AC03-76SF00098 through an LDRD grant and NSF Grant CCR-9001986.

References

[1] P.K. Agarwal and C.M. Procopiuc. Exact and approximation algorithms for clustering. *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 658–667, 1998.

[2] D. Boley. Principal direction divisive partitioning. *Data mining and knowledge discovery*, 2:325–344, 1998.

[3] B. Bollobas. *Random Graphs*. Academic Press, 1985.

[4] C.-K. Cheng and Y.A. Wei. An improved two-way partitioning algorithm with stable performance. *IEEE. Trans. on Computed Aided Desgin*, 10:1502–1511, 1991.

[5] F.R.K. Chung. *Spectral Graph Theory*. Amer. Math. Society, 1997.

[6] C. Ding, X. He, and H. Zha. A spectral method to separate disconnected and nearly-disconnected web graph components. *Proc. 7th ACM Int'l Conf Knowledge Discovery and Data Mining (KDD 2001)*, pages 275–280, August 2001.

[7] S. Doddi, M.V. Marathe, S. S. Ravi, D. S. Taylor, and P. Widmayer. Approximation algorithms for clustering to minimize the sum of diameters. *Nordic Journal of Computing, 7(3):185, Fall 2000*, Fall 2000.

[8] W.E. Donath and A. J. Hoffman. Lower bounds for partitioning of graphs. *IBM J. Res. Develop.*, 17:420–425, 1973.

[9] R. V. Driessche and D. Roose. An improved spectral bisection algorithm and its application to dynamic load balancing. *Parallel Computing*, 21, 1995.

[10] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering in large graphs and matrices. In *Proc. 19th ACM-SIAM Symposium on Discrete Algorithms*, 1999.

[11] M. Fiedler. Algebraic connectivity of graphs. *Czech. Math. J.*, 23:298–305, 1973.

[12] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czech. Math. J.*, 25:619–633, 1975.

[13] S. Guattery and G. L. Miller. On the quality of spectral separators. *SIAM Journal of Matrix Anal. Appl.*, 19(3), 1998.

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

[15] X. He, H. Zha, C. Ding, and H.D. Simon. Web document clustering using hyperlink structures. *Tech Report CSE-01-006*, April 2001.

[16] B. Hendrickson and R. Leland. Chaco mesh partitioning software. *http://www.cs.sandia.gov/CRF/chac.html*.

[17] G. Karypis and V. Kumar. Metis graph partitioning software. *http://www-users.cs.umn.edu/ karypis/metis/*.

[18] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Tech. J.*, 1970.

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

[20] B. N. Parlett. *The Symmetric Eigenvalue Problem*. SIAM Press, 1998.

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

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

[23] D.A. Spielman and S.-H. Teng. Spectral partitioning works: Planar graphs and finite element meshes. *Proc. 37th IEEE Conference on Foundations of Computer Science,*, 1996.

[24] H. Zha, X. He, C. Ding, M. Gu, and H.D. Simon. Bipartite graph partitioning and data clustering. *Proc. 10th Int'l Conf. Information and Knowledge Management (CIKM 2001)*, Nov. 2001.