CS 395T	Large-Scale Data Mining	Fall 2001

Lecture 10

Lecturer: Inderjit Dhillon Date: Oct. 23, 2001

Keywords: *Graph Partitioning, Kernighan-Lin, Spectral Partitioning, Multilevel Methods*

1 Objective

Given a graph G = (V,E) where V is the set of vertices and E is the set of edges, the informal objective of graph partitioning is to partition V into disjoint parts V_1 and V_2 such that

1. Number of crossing edges between V_1 and V_2 is minimized.
2. V_1 and V_2 are nearly equal in size.

The desire to get equal-sized partitions makes the problem hard – it is much harder than the min-cut problem. In fact, the graph partitioning problem is NP-complete. One formal objective, the ratio cut objective, is:

$$\min_{V_1,V_2} \frac{Cut(V_1,V_2)}{|V_1|} + \frac{Cut(V_1,V_2)}{|V_2|}$$

Note that if two partitionings have the same cut value, the above objective favors the more equal-sized partition. Many heuristic algorithms have been proposed for the problem of graph partitioning. Three of the most successful algorithms are given in the next three sections.

2 Kernighan-Lin (KL) Algorithm

The Kernighan-Lin algorithm is a local descent algorithm, and it produces equal (or off-by-one) partitions. The algorithm starts with an edge weighted graph

$$G = (V, E, W_E)$$

and a partitioning

$$V = A \cup B, \quad |A| = |B|$$

Goal : Find X $\subset$ A, Y $\subset$ B, and swap(exchange) X and Y such that this swap reduces the total cost of edges from A to B.

$$Current\ cost\ = \sum_{a\in A, b\in B} w(a,b),$$

where $w(a,b)$ is the weight of edge (a,b).
Suppose $A_{new} = (A - X) \cup Y$, $B_{new} = (B - X) \cup Y$, then

$$New\ cost\ = \sum_{a\in A_{new}, b\in B_{new}} w(a,b)$$

To compute the new cost efficiently, for $a \in A$ maintain

E(a) = external cost of $a = \sum_{b \in B} w(a,b)$
I(a) = internal cost of $a = \sum_{a' \in A, a' \neq a} w(a,a')$
C(a) = E(a) - I(a)

and analogously for $b \in B$ maintain

E(b) = external cost of b $= \sum_{a \in A} w(a,b)$
I(b) = internal cost of b $= \sum_{b' \in B, b' \neq b} w(b,b')$
C(b) = E(b) - I(b)

If we swap $a \in A$ and $b \in B$ then

$$\begin{aligned} NewCost &= OldCost + [I(a) - (E(a) - w(a,b))] + [I(b) - (E(b) - w(a,b))] \\ &= OldCost - [C(a) + C(b) - 2w(a,b)] \\ &= OldCost - Gain(a,b) \end{aligned}$$

Note that, in general, $C(a')$ and $C(b')$ also change for all $a' \in A$, $a' \neq a$ and $b' \in B$, $b' \neq b$.

2.1 The Algorithm

1. Initialize partitions A & B and compute total *Cost*.
2. Compute C(x) for all vertices x.
3. Unmark all vertices. Set $i = 1$.
4. While there are unmarked vertices do
   (a) Find the unmarked pair(a_i, b_i) that maximizes $Gain(a_i, b_i)$.
   (b) Mark a_i and b_i (don't swap).
   (c) Update C(x) for all unmarked x, as though a_i & b_i had been swapped.
   (d) i $\leftarrow$ i + 1.
5. Pick j that maximizes Gain $= \sum_{i=1}^{j} Gain(a_i, b_i)$.
6. If Gain > 0 then update

$$\begin{aligned} A &= (A - \{a_1, \ldots a_j\}) \cup \{b_1, \ldots b_j\}, \\ B &= (B - \{b_1, \ldots b_j\}) \cup \{a_1, \ldots a_j\}, \\ Cost &= Cost - Gain. \end{aligned}$$

7. If Gain > 0, go to step 2.

Step 4 is executed $|V|/2$ times during each iteration while step $4(a)$ requires $O(|V|^2)$ time. Thus, the total time taken by the algorithm is $O(|V|^3)$ assuming number of iteration is a fixed constant. Note that some Gain(a_i, b_i) may be negative. This makes it possible to escape some "local minima". But overall, KL is still a local algorithm. A better implementation by Fiduccia & Matthyses(FM), 1982, costs $O(|E|)$ per iteration.

3 Spectral Partitioning

Consider the node-edge incidence matrix $\mathbf{E}$ where $\mathbf{E}$ is a $|V| \times |E|$ matrix where the column corresponding to edge $e = \{i, j\}$ is zero except for the i^{th} and j^{th} entries which are +1 or -1 respectively. Let $\mathbf{L} = \mathbf{E}\mathbf{E}^T$ which is a $|V| \times |V|$ matrix. It is easily seen that $\mathbf{L}$(i,i) = degree of node i, and $\mathbf{L}$(i,j) = -1, if there is an edge $\{i, j\}, i \neq j$. Since $\mathbf{L} = \mathbf{E}\mathbf{E}^T$ it is (a) symmetric, (b) positive semi-definite matrix $\Rightarrow$ all eigenvalues of $\mathbf{L}$ are real and non-negative. It is easy to see that $\mathbf{Le} = 0$, where $\mathbf{e} = [1, 1, \ldots, 1]^T$, i.e., $\mathbf{e}$ is the vector of all ones. Thus $\mathbf{L}$ has a zero eigenvalue. The matrix $\mathbf{L}$ is called the Laplacian of the graph.

Many graph properties can be expressed in terms of the eigenvalues of $\mathbf{L}$. For example,

- number of zero eigenvalues of $\mathbf{L}$ = number of connected components of G.
- Let λ and $\mathbf{v}$ be an eigenvalue & normalized eigenvector of $\mathbf{L}$ respectively. Then

$$\mathbf{Lv} = \lambda\mathbf{v} \;\Rightarrow\; \mathbf{v}^T\mathbf{Lv} = \lambda$$

Suppose G is connected, then $\lambda_2(G)$ (second smallest eigenvalue of $\mathbf{L}$) is positive. Let V_1 and V_2 be the vertex partitions of V and let the vector $\mathbf{x}$ capture this partitioning information,

$$x_i = \begin{cases} 1 & \text{if node i} \in V_1 \\ -1 & \text{if node i} \in V_2 \end{cases}$$

Then

$$\begin{aligned} \mathbf{x}^T\mathbf{Lx} = \mathbf{x}^T\mathbf{E}\mathbf{E}^T\mathbf{x} &= \sum_{(i,j)\in E} (x_i - x_j)^2 \\ &= 4 \cdot Cut(V_1, V_2) \end{aligned}$$

Thus we are looking for a ± 1 vector $\mathbf{x}$ such that $\mathbf{x}^T\mathbf{Lx}$ is minimized. This problem is still NP hard, but its real relaxation (i.e., relax the ± 1 constraint on $\mathbf{x}$) implies that the optimal $\mathbf{x} = \mathbf{v}_2$ = eigenvector corresponding to λ_2 (note that the constraint on the size of the vertex partitions implies that $\mathbf{x}^T\mathbf{e} = 0$, i.e., $\mathbf{x}$ is orthogonal to the eigenvector $\mathbf{e}$ that corresponds to the smallest eigenvalue 0).

3.1 Simple Spectral Bisection Algorithm

- Compute eigenvector $\mathbf{v}_2$ for eigenvalue λ_2 of L(G).
- For each node i $\in$ V,
  - If $\mathbf{v}_2$(i) < 0
    - put node i in partition V_1
  - else
    - put node i in partition V_2
  - end if

Many other extensions to this simple algorithm have been studied. Spectral partitioning gives a good global heuristic that can be improved by the "local-smoothing" of the KL or FM algorithms. For an application to document and word clustering (and more theoretical details), see [Dhi01].

4 Multilevel Methods

- Very Fast.
- Good partitioning results.
- For example, METIS, CHACO software.
- Multilevel KL or multilevel spectral.
- Multilevel methods use a divide and conquer approach.

4.1 Steps

1. Coarsen the graph G.
2. Partition coarse graph(may do this by recursively going to step 1).
3. Refine the coarse partitioning to get partitioning for G.

For details on Metis, see [KK99].

References

[Dhi01] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the The Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2001)*, 2001. also appears as UT CS Technical Report # TR 2001-05, March, 1999.

[KK99] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1999.