CS 7880 Network Algorithms and Analysis **Spring 2012**

Lecture 04/13: Graph Sampling and Sparsification

Lecturer: Rajmohan Rajaraman *Scribes: Qian Zhang*

1 Backgrounds

In previous lectures, we discussed various topics on graph: sparsest cut, clustering which finds small k-cuts, max flow, multicommodity flow, etc. For very large graphs, we'd like to ask "instead of regular approximation algorithms which run in polynomial time, can we find any near-linear time algorithm for minimum cut or maximum flow problems?". A simple idea is for a given graph G, to solve the problem on a sparse subgraph with the same number of nodes of G that preserves some properties of the original graph like distance or cuts, rather than on the original graph, and formally is called graph sparsification with sampling.

2 Graph Sparsification Maintaining Cuts (Karger's Basic Min-Cut)

Karger's basic min-cut algorithm tries to solve min-cut problems by solving $n-1$ $s-t$ min-cut problems in near-linear time. The basic idea is based on the concept of contraction of an edge in an undirected graph G. Informally, the contraction makes the two nodes connected by the edge overlap, and thereby reduces the total number of nodes in the graph by one. The algorithm is based on sequences of contractions of a randomly chosen edge.

Let H be the sampled graph of G, and G is partitioned into two subgraphs A and B by the cut c. Suppose we choose each edge with probability p, so the expectation size of cut in H $E[c_H(A,B)] = p \cdot c_G(A,B)$. If we fix the cut of size S in G, the expected size of cut in H is $E[\text{size of cut in } H] = pS$. Suppose the edges in the cut c are $e_1, e_2, \ldots, e_l$, and define random variable $x_i = 1$ if the edge e_i is selected and $x_i = 0$ if not. So the random variable $X = \sum x_i$ equals to the size of the cut in H, and its expectation $\mu = E[X] = pS$. From Chernoff-Hoeffding Bounds[1], we have

$$\Pr[|X-\mu| \geq \varepsilon\mu] \leq 2\exp(-\frac{\varepsilon^2\mu}{3}) = 2\exp(-\frac{\varepsilon^2 pS}{3})$$

So if we want $\exp(-\frac{\varepsilon^2 pS}{3})$ to be small, we need a larger value of p, but when p is large, like close to 1, the running time grows. The question now is how to choose a proper value of p?

[1]Chernoff-Hoeffding Bounds: Suppose $x_1, x_2, \ldots, x_l \in [0,1]$ are random independent variables, and $X = \sum_{i=1}^{l} x_i$, its expectation $\mu = E[X]$. There is

$$\Pr[|X-\mu| \geq \varepsilon\mu] \leq 2\exp(-\frac{\varepsilon^2\mu}{3})$$

We would like to find the min-cut in time $O(mc^*)$, where m is the number of edges in G and c^* is the size of min-cut. If c^* is small ($O(\log n)$ where n is the number of vertices in G), the running time is $O(m \log n)$. But when the cut size is large ($\Omega(\log n)$), we show with a proper choice of p, the min-cut in the sample graph can be found in similar time complexity.

Suppose $S = \ln n$ and the min-cut size c^* is known ($c^* \leq S = \ln n$), and let

$$p = \frac{d \ln n}{\varepsilon^2 c^*}$$

where d and ε are constants, so

$$\begin{aligned} \Pr[|\text{cut-size} - pS| \geq \varepsilon pS] &\leq 2\exp(-\frac{\varepsilon^2 d \ln nS}{3\varepsilon^2 c^*}) \\ &\leq \frac{2}{n^{dS/3c^*}} \leq \frac{2}{n^{d/3}} \end{aligned}$$

when n is very large and with a proper choice of d, like $d = 2$, such the probability can be sufficiently small. Let the number of edges in H be m_H and the size of min-cut in H be c^*_H, there are

$$m_H \approx pm = \frac{md \ln n}{\varepsilon^2 c^*} \quad \text{and} \quad c^*_H \approx pc^* = \frac{c^* d \ln n}{\varepsilon^2 c^*} = \frac{d \ln n}{\varepsilon^2}$$

So the min-cut in H can be found in time

$$O(m_H c^*_H) = O(\frac{md \ln n}{\varepsilon^2 c^*} \cdot \frac{d \ln n}{\varepsilon^2}) \leq O(\frac{d}{\varepsilon^2} m \ln n)$$

which means with a proper choice of p, the min-cut of the sampled graph can also be found in $O(m \ln n)$[2]. Below and in the next lecture, we will show the correctness of the algorithm.

Formally, let $G = (V, E)$ be an undirected graph, and $n = |V|$, $m = |E|$. For $S \subset V$, the set $c(S, V - S) = \{(u, v) \in E : u \in S, v \in V - S\}$ is a cut of G. The concept of contraction can be defined as the following, and illustrated in Figure 1[3].

Definition 1 *Let $G = (V, E)$ be a graph without self-loop. For $e = (u, v) \in E$, the contraction with respect to e, denoted $G = G/e$, is defined by:*

1. *Replace vertices u and v with a new vertex x.*
2. *Replace all edges (u, i) or (v, i) with an edge (x, i).*
3. *Remove self-loops to x.*

The key observation here is that if we contract an edge (u, v) then we preserve those cuts where u and v are both in S or in $V - S$.

[2]Note in fact the probability of selecting an edge can be different for edges. For example, if an edge is a part of a large cut, the probability for this edge should be small, and vice-versa for small cuts, which suggests that the probability for each edge should be inversely proportional to some quantity correlated with the connectivity.

[3]The figure is extracted from wikipedia

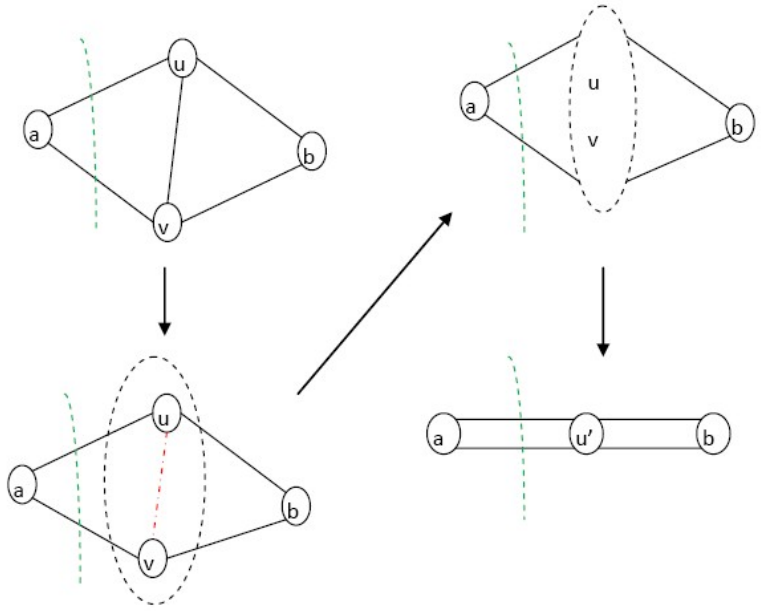

Figure 1: The process of contraction.

Observation 2 *Let $e = (u, v) \in E$. There is a one-to-one correspondence between cuts in G which do not contain any edge (u, v), and cuts in G/e. In fact, for $S \subset V$ such that $u, v \in S$, $c_G(S, V - S) = c_{G/e}(S, V - S)$.*

The idea of the algorithm is to contract $n - 1$ edges and then two vertices remain. These two vertices correspond to a partition $(S, V - S)$ of the original graph, and the edges remaining in the two vertex graph correspond to $c(S, V - S)$ in the original graph. We'd like to output this cut as the minimum cut of the original graph. So the question now is "what edges do we contract?". If we never contract edges from the minimum cut $c(S^*, V - S^*)$, from the above observation, this is the cut we are looking for. Since $c(S^*, V - S^*)$ is of minimum size, it has relatively few edges, so if we contract a random edge it turns out that we will have a reasonable probability of preserving this min-cut. The formal algorithm is defined as following.

Algorithm 3 *Karger's basic min-cut algorithm:*

Starting from the input graph $G = (V, E)$, repeat the following process until only two vertices remain:

1. Pick an edge $e = (u, v)$ uniformly at random from E
2. Contract (u, v), i.e., set $G = G/e$

These final two vertices correspond to set S and $V - S$ in the original graph, and the edges remaining in the final graph correspond to the edges in the cut $c(S, V - S)$ of the original graph.

Claim 4

$$\Pr[\textit{min-cut surviving until end}] \geq \frac{1}{\binom{n}{2}}$$

Proof: Fix a minimum cut c^* in the original graph, we know the degree of each node is at least c^*, and the number of edges in contracted graph is at least $(n' - c^*)/2$, where n' is the number of nodes in the contracted graph. So

$$\Pr[\text{min-cut surviving in that round}] \geq 1 - \frac{2}{n'}$$

and

$$\begin{aligned} \Pr[\text{min-cut surviving until end}] &\geq (1 - \frac{2}{n})(1 - \frac{2}{n-1}) \dots (1 - \frac{2}{3}) \\ &= \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}} \end{aligned}$$

■