15-859(M): Randomized Algorithms **Lecturer:** Anupam Gupta
Topic: Randomized Min Cut **Date:** September 27th, 2004
Scribe: Vineet Goyal

5.1 Introduction

Given an undirected, weighted graph $G = (V, E)$, and two vertices s and t, an $s-t$ cut is a partition of vertices into two parts $[S, T]$ such that s and t are on different sides of the partition. We can also refer to the cut as the edges going across the partition $[S, T]$ and cost of the cut is the sum of costs of these edges. The problem of minimum $s-t$ cut can be solved using the duality between max flows and min cut. In this lecture, we will see how to find a global min cut i.e $\min_{s,t \in V} \text{cost}(s-t \text{ cut})$.

A naive approach would be to find $s-t$ min cuts for all choices of s and t and output the minimum cut found. This would take C_2^n flow computations. We can do slightly better. Fix any arbitrary vertex s. Now, compute $s-t$ min cuts for all choices of t. Clearly, s will be on one side of the partition in the global min cut. Thus, for any vertex t on the other side of partition, $s-t$ min cut will be the global min cut. Therefore, computing $s-t$ min cuts for all choices of t will give us the global min cut. But still we require $(n-1)$ max flow computations.

5.2 A Randomized Algorithm

In this lecture, we will look at a simple randomized algorithm due to Karger [1] that gives a global min cut with high probability. For simplicity, we consider an unweighted graph. However, it can be easily generalized to the weighted case.

Algo Min-cut
$i \leftarrow 0$, $G_0 \leftarrow G$
while (G_i has more than two vertices), do
  Pick an edge e, uniformly at random (u.a.r) from G_i.
  $G_{i+1} \leftarrow$ Contract e in G_i and remove self loops.
  $i \leftarrow i + 1$
od
Output the edges going between 2 vertices left in G_i.

Note that, when we contract an edge e, apart from the self loops(which we remove), we can also get parallel edges. Thus, the graph obtained after contraction is a multigraph with no self loops. Let us now prove the correctness of the algorithm.

Proposition 5.2.1 *Every cut in contracted graph at step i corresponds to some cut in graph at step $j < i$.*

Consider a cut C in the graph G_i. If we uncontract the edge contracted in G_{i-1} to obtain G_i, we obtain a corresponding cut C' in the graph G_{i-1}. Note that C and C' contain exactly the same edges as the uncontracted edges in G_{i-1} are on the same side of the cut C.

Proposition 5.2.2 *Suppose the min-cut in G is of size k, then size of min-cut in G_i is greater than or equal to k.*

Any cut in G_i corresponds to a cut in G. In particular, min-cut in G_i also corresponds to some cut in G. Thus, size of min-cut in G_i is greater than or equal to k.

Proposition 5.2.3 *G_i has at least $(n-i).\frac{k}{2}$ edges.*

Since the size of min-cut in $G_i \geq k$, the degree of each vertex in G_i is $\geq k$. G_i has $(n-i)$ vertices. Thus, G_i has at least $(n-i).\frac{k}{2}$ edges.

Proposition 5.2.4 *Suppose $C \subseteq E$ is a min-cut in G. If none of the edges in C are picked for contraction $\Rightarrow$ we will output C.*

If none of the edges of C get contracted, then clearly we will output C. An edge e can be contracted if either it is picked for contraction or some edge parallel to e is picked for contraction. We know that none of the edges of C were picked for contraction. Thus, the only way an edge $e \in C$ can be contracted is if we pick some edge parallel to e. But note that if $e \in C$, then any edge parallel to e is also in C. Hence, e can't be contracted and we will output C.

For the sake of analysis, let us fix our favourite min-cut in G, say C.

Claim 5.2.5 $\mathbf{Pr}[C$ *survives algorithm* $] \geq \frac{1}{{}^nC_2}$.

Proof:

$$\mathbf{Pr}[C \text{ survives algorithm }] = \mathbf{Pr}[C \text{ survives when graph has 2 vertices }]$$

Let Event $E_i = \{\text{cut } C \text{ survives the iteration when graph has } i \text{ vertices}\}$.
We need to calculate $\mathbf{Pr}[E_3]$.

$$\begin{aligned}
\mathbf{Pr}[E_3] &= \mathbf{Pr}[E_3|E_4] \cdot \mathbf{Pr}[E_4] + \mathbf{Pr}[E_3|\neg E_4] \cdot \mathbf{Pr}[\neg E_4] \\
&= \mathbf{Pr}[E_3|E_4] \cdot \mathbf{Pr}[E_4] + 0 \\
&= \mathbf{Pr}[E_3|E_4] \cdot \mathbf{Pr}[E_4|E_5] \ldots \mathbf{Pr}[E_{n-1}|E_n] \cdot \mathbf{Pr}[E_n]
\end{aligned}$$

$$\begin{aligned}
\mathbf{Pr}[\neg E_n] &= \frac{|C|}{|E(G)|} \\
&\leq \frac{k}{k.n/2} = \frac{2}{n}
\end{aligned}$$

$$\mathbf{Pr}[E_n] = \frac{n-2}{n}$$

Similarily,

$$\mathbf{Pr}[\neg E_i|E_{i+1}] \leq \frac{k}{k.i/2} = \frac{2}{i}$$

$$\mathbf{Pr}[E_i|E_{i+1}] \geq \frac{i-2}{i}$$

Hence,

$$\mathbf{Pr}[E_3] \geq \frac{2}{(n-1).n} = \frac{1}{{}^nC_2}$$

■

Thus, we have proved that the probability of success (i.e. cut C surviving the algorithm) is at least $\frac{1}{{}^nC_2}$. We can boost the probability of success by doing more independent runs of the algorithm. Suppose we repeat the algorithm t times.

$$\mathbf{Pr}[\text{Failiure after t trials}] = (1 - \mathbf{Pr}[\text{success}])^t \leq e^{-\mathbf{Pr}[\text{success}] \cdot t}$$

Thus, if we choose $t = \frac{c \log n}{\mathbf{Pr}[\text{success}]}$,

$$\mathbf{Pr}[\text{Failiure after t trials}] \leq \frac{1}{n^c}$$

Hence, we get a small probability of failiure and whp we output a min-cut of G. From Claim 5.2.5, we obtain the following interesting observations.

Observation 1: There are at most ${}^nC_2$ min-cuts in any graph.
Since, the probability that any particular min-cut C survives, is at least $\frac{1}{{}^nC_2}$ and no two cuts can survive simultaneously. Thus, there can be at most ${}^nC_2$ min-cuts in any graphs.

Observation 2: The probability of success $\frac{1}{{}^nC_2}$ is tight. Cycle of length n is a tight example for this.

The algorithm described above works for the case of unweighted, undirected graphs. But it is quite straightforward to extend this algorithm so that it works for weighted graphs. Instead of choosing an edge e for contraction, uniformly at random, we choose an edge e with probability $\frac{w(e)}{\sum_{e' \in E} w(e')}$. A similar analysis gives us exactly the same bounds on probability of success.

5.3 Modified Randomized Algorithm

Note that, we will need $\tilde{O}(n^2)$ runs of the algorithm and in each run of the algorithm, we do $n-1$ contractions. Thus, we have $\tilde{O}(n^3)$ contractions in all. We will describe a modified algorithm due to Karger and Stein [2] that requires $\tilde{O}(n^2)$ contractions.

Modified Algo Min-Cut

1. Run *Algo Min-Cut* on G until we obtain a graph G_1 with $\frac{n}{\sqrt{2}}$ vertices left.
2. Run *Algo Min-Cut* again on G independently, to obtain a graph G_2 with $\frac{n}{\sqrt{2}}$ vertices.
3. $C_1 \leftarrow$**Modified Algo Min-Cut**(G_1).

4. $C_2 \leftarrow$**Modified Algo Min-Cut**(G_2).

5. Output the better of two cuts C_1 and C_2.

The above algorithm performs $O(n^2)$ contractions. Let $T(n)$ denote the number of contractions for a graph of size n. Therefore, $T(n) = 2T(\frac{n}{\sqrt{2}}) + O(n)$, and $T(2) = 0$. Thus, $T(n) = O(n^2)$. Now, we prove that the probability of our favourite min cut surviving the algorithm is at least $\frac{1}{2 \cdot \log n}$. First let us calculate the probability that C survives in G_1. By a method similar to one used in the calculation of probability of success in Algo Min-Cut, we get that $\mathbf{Pr}[C \text{ survives in } G_1] = \frac{1}{2}$. Similarly, $\mathbf{Pr}[C \text{ survives in } G_2] = \frac{1}{2}$. Consider the recursion tree of the modified algorithm. Let p_d denote the probability that C survives a recursion subtree of depth d given that C survived till the root. Therefore,

$$p_d = \frac{1}{2} \cdot p_{d-1} + \frac{1}{2} \cdot p_{d-1} - \frac{1}{4} \cdot p_{d-1}^2$$

Claim 5.3.1 $p_d \geq \frac{1}{d+1}$

Proof: We prove this claim by induction.
Base Case : $d = 0$. Trivially true, as given that C survives at the root of a depth 0 subtree, probability it survives the subtree is 1.
Induction Step : $d \geq 1$.

$$p_d = p_{d-1} - \frac{p_{d-1}^2}{4} = \frac{1}{d} - \frac{1}{4d^2} \geq \frac{1}{d+1}$$

Hence, proved. ∎

The depth of the recursion tree is $log_{\sqrt{2}} n$. Thus,

$$\mathbf{Pr}[C \text{ survives the algorithm}] \geq \frac{1}{2 \log n + 1}$$

Therefore, if we run the algorithm $O(\log n)$ times independently, we obtain the min-cut with very high probability and the number of contractions is only $\tilde{O}(n^2)$.

References

[1] D.R. Karger. Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 21-30, 1993.

[2] D.R. Karger and C. Stein. An $\tilde{O}(n^2)$ algorithm for minimum cuts. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 757-765, 1992.