Massachusetts Institute of Technology | Handout 65
6.042J/18.062J: Mathematics for Computer Science | June 8, 1998
Professor Albert R. Meyer

The Maximum Cut Problem

We illustrate the application of Chebyshev's Theorem in the analysis of a randomized algorithm for the Maximum Cut problem.

1 The Problem

Suppose $G = (V, E)$ is an undirected simple graph and $S \subseteq V$ is a subset of the vertices. So S and its complement $V - S$ partition the vertices into two parts. An edge $e \in E$ is said to *cross* the partition iff one of the endpoints of e is in S and the other is in $V - S$. We also say that the partition *cuts* the edge when the edge crosses the partition. For any $S \subseteq V$, let $\text{Cut}_G(S)$ be the number of edges that cross the partition $(S, V - S)$.

The *Maximum Cut Problem* is, given G, to partition the vertices to maximize the number of edges that cross the partition, that is, to find S such that $\text{Cut}_G(S)$ is a maximum.

A simple way of optimally solving Max Cut is to cycle through all the possible partitions and then return the best one. Unfortunately, this exhaustive algorithm is impractical even for fairly small graphs because the time it takes grows exponentially: it requires examination of $2^{|V|}$ partitions. There are good theoretical grounds to believe that there is no significantly better optimal algorithm, i.e., that every optimal algorithm for Max Cut must have a running time growing exponentially in the number of vertices, at least in the worst case. However, if we settle for *approximately*, but not exactly, optimal cuts, it is possible to find efficient algorithms which cut a lot of edges. In the next section we describe a simple-minded, efficient randomized algorithm which does this.

2 The Algorithm

Consider the following randomized algorithm for the Maximum Cut problem: initially let S be empty; independently for each vertex $v \in V$ flip a coin, and add v to S when the coin shows heads.

This algorithm may not seem to make much sense. First of all the algorithm *does not even look at the graph*, but just makes random and independent choices for each vertex. So each of the subsets of V is equally likely to be chosen as S. This means the worst possible partition, namely $(\emptyset, V)$, is as likely to be generated as any other partition. Despite these objections, we will see that this simple randomized algorithm has very good typical behavior.

3 Some Random Variables

Define for every vertex $v \in V$ a Bernoulli random variable X_v defined as follows:

$$X_v = \begin{cases} 1 & \text{if } v \in S, \\ 0 & \text{otherwise.} \end{cases}$$

The X_vs are mutually independent and unbiased by the definition of our algorithm.

We also introduce a random variable Y_e for every edge $e \in E$, defined as follows:

$$Y_e ::= \begin{cases} 1 & \text{if } e \text{ crosses } (S, V - S), \\ 0 & \text{otherwise.} \end{cases}$$

Notice that

$$\mathrm{Cut}_G(S) = \sum_{e \in E} Y_e.$$

Lemma 3.1. The random variables Y_e are unbiased.

Proof. We use the definition of crossing and the independence of the X_vs. Let $\{u, v\} \in E$, then

$$\begin{aligned}
\Pr\left[Y_{\{u,v\}} = 1\right] &= \Pr\left[(X_u = 1 \wedge X_v = 0) \vee (X_u = 0 \wedge X_v = 1)\right] \\
&= \Pr\left[X_u = 1 \wedge X_v = 0\right] + \Pr\left[X_u = 0 \wedge X_v = 1\right] \\
&= \Pr\left[X_u = 1\right] \Pr\left[X_v = 0\right] + \Pr\left[X_u = 0\right] \Pr\left[X_v = 1\right] \\
&= \frac{1}{2}\frac{1}{2} + \frac{1}{2}\frac{1}{2} \;=\; \frac{1}{2}.
\end{aligned}$$

□

Lemma 3.2. The variables Y_e are pairwise independent.

Proof. Consider two edges e_1 and e_2 in E. If the two edges do not share any vertex, then the independence of Y_{e_1} and Y_{e_2} follows from the independence of their endpoints.

The case to consider is when they share an endpoint, namely $e_1 = \{u, v\}$ and $e_2 = \{v, w\}$. Then, for example,

$$\begin{aligned}
\Pr[Y_{e_1} = 1 \wedge Y_{e_2} = 1] &= \Pr[X_u \neq X_v \wedge X_v \neq X_w] \\
&= \Pr[(X_u = X_w = 0 \wedge X_v = 1) \vee (X_u = X_w = 1 \wedge X_v = 0)] \\
&= \Pr[X_u = X_w = 0 \wedge X_v = 1] + \Pr[X_u = X_w = 1 \wedge X_v = 0] \\
&= \frac{1}{8} + \frac{1}{8} = \frac{1}{4} \\
&= \Pr[Y_{e_1} = 1] \Pr[Y_{e_2} = 1],
\end{aligned}$$

and similar calculations work for every other value of Y_{e_1} and Y_{e_2}. □

4 The Analysis

By Lemma 3.1, $\mathrm{E}[Y_e] = 1/2$. Using the linearity of expectation, we conclude that

$$\mathrm{E}[\mathrm{Cut}_G(S)] = \mathrm{E}\left[\sum_{e \in E} Y_e\right] = \sum_{e \in E} \mathrm{E}[Y_e] = |E|/2.$$

In other words, our algorithm cuts on the average half the edges. (It is an interesting corollary of this calculation that in every graph there *always exists* a vertex partition that cuts at least half the edges.)

The next step of our analysis consists in bounding the probability that our algorithm performs much worse than its expected behavior.

By Lemma 3.2, the random variable $\mathrm{Cut}_G(S)$ is a sum of pairwise independent unbiased Bernoulli variables, and so

$$\mathrm{Var}[\mathrm{Cut}_G(S)] = \mathrm{Var}\left[\sum_{e \in E} Y_e\right] = |E| \,\mathrm{Var}[Y_e] = \frac{|E|}{4}.$$

Using the Chebyshev bound, we can now estimate the probability that the number of cut edges differs much from $|E|/2$. Namely, for every constant $c > 0$,

$$\Pr\left[|\mathrm{Cut}_G(S) - |E|/2| \geq c\sqrt{|E|}\right] \leq \frac{1}{4c^2}.$$

For example, there is less than a 0.1% chance that the number of cut edges deviates from the expectation by more than an additive term $8\sqrt{|E|}$—a very small term compared to $|E|$ when there are a lot of edges.

Since our goal is to maximize the number of cut edges, we can use the stronger estimate given by the one-sided Chebyshev Theorem,

$$\Pr\left[\mathrm{Cut}_G(S) \leq |E|/2 - c\sqrt{|E|}\right] \leq \frac{1}{1+4c^2}.$$

We remark that although the variables Y_e are pairwise independent, they are not even three-way independent. For example, if edges e_1, e_2, e_3 form a triangle, then $(Y_{e_1} + Y_{e_2}) \bmod 2 = Y_{e_3}$, so the values of any two of the variables $Y_{e_1}, Y_{e_2}, Y_{e_3}$ uniquely determine the value of the third. That's why the Chernoff bound, which requires full independence, is not applicable.

5 How Close to Optimal?

Since the random algorithm above produces a partition that cuts at least half the edges, it will, with high probability, inevitably cut at least 50% of the number of edges which can optimally be cut. This is the best percentage of optimal that can be guaranteed for this algorithm. For example, on a bipartite graph with an equal number of vertices in each half, the algorithm will cut on the average half of the edges, whereas partitioning into the two halves of the bipartite graph cuts *all* the edges.

The algorithm above does come close to optimal with very high probability on cliques, that is, graphs with edges between every two vertices. Namely, for a clique with $2n$ vertices, the best partition divides the vertices into two blocks of n, so that n^2 of the $n(2n-1)$ edges are cut. Our randomized algorithm cuts on the average $n(2n-1)/2 = n^2 - n/2$ edges, and cuts with high probability $n^2 - O(n)$ edges. So in this case the algorithm can be expected to cut nearly 100% of the optimal number of edges as n increases.

An algorithm essentially equivalent to the random one above was proposed in 1974. Despite much effort, no efficient algorithm guaranteeing a better ratio to the optimum had been found until 1994, when Goemans and Williamson at MIT found an efficient algorithm for cutting at least 87.8% of the optimum number of edges. This percentage is the best which has been achieved to date. It is also known that finding cuts within a factor $16/17 \approx 94.1\%$ of the optimum is as hard as solving the problem optimally, so there is not much room for improvement on the Goemans and Williamson percentage.

These notes are by Luca Trevisan.