6.891 Approximation Algorithms	February 25, 2000
Lecture 4	
Lecturer: David Karger	*Scribe: Andrew Menard*

1 Randomized Methods

Today we will consider the following problem.

MAX CUT

- **Input:** An undirected graph $G = (V, E)$
- **Goal:** A partition $V = V_L \cup V_R$, so as to maximize $|E \cap (V_L x V_R)|$ (that is, the number of edges with one endpoint on each side of the partition).

You could also add non-negative weights to the edges, and have the goal change to maximizing the weight of the edges crossing the cut; all of the algorithms here still apply, and still have the same performance.

This is an NP-complete problem, and for a long time the best known approximation algorithm was a $\frac{1}{2}$-approximation. In 1994, Goemans and Williamson used semidefinite programming to get a .87856-approximation.

MAX-CUT has also been shown to be MAX-SNP-hard, which means there is a bound on how closely it can be approximated. Thus, there is no PTAS in general for this problem.

Result for today will be PTAS for MAX-CUT on dense graphs.

1.1 A simple algorithm for MAX CUT

We start with simple algorithms. The simplest is random placement of vertices, which generates a $\frac{1}{2}$-approximation. To show this, consider the probability that any given edge $e = (v_1, v_2)$ crosses the cut. This probability is clearly $\frac{1}{2}$, since whichever side v_1 is on, v_2 is on the opposite side with probability $\frac{1}{2}$. If a graph has m edges, then the expected number of edges cut is $\frac{m}{2}$ by linearity of expectation.

We know that $OPT \leq m$, since the maximum possible cut would cut evey edge, so the random placement of vertices is a $\frac{1}{2}$-approximation.

Applying derandomization to this generates a greedy algorithm: place each vertex so as to maximize the number of edges cut. For each vertex, you cut at least $\frac{1}{2}$ of the set of edges connecting this vertex to previous vertices. Since every edge is in exactly one such set, overall at least half of all the edges are cut, so you again have a $\frac{1}{2}$-approximation algorithm.

Thinking about this also gives a local optimality condition; every vertex has at least $\frac{1}{2}$ of its neighbors on the opposite side in an optimal solution. Otherwise, you could swap this vertex to the other set and increase the value of the solution.

1.2 Random sampling

The next technique is random sampling. The idea is simple – take a large problem, and sample from it to get a representative subproblem which is small enough to be easily solvable and representative enough to give you useful information about solving the original problem.

An extremely useful theorem is the following, often called a *Chernoff bound.*

Theorem 1 (Chernoff bound) *Let $X = \sum_i X_i$ whiere each X_i is a $0-1$ random variable with any distribution. Let $\mu = E(X)$. Then*

$$\Pr(X \leq (1-\epsilon)\mu) \leq e^{\frac{-\epsilon^2\mu}{2}}$$

and

$$\Pr(X \geq (1+\epsilon)\mu) \leq e^{\frac{-\epsilon^2\mu}{3}}$$

for $0 \leq \epsilon \leq 1$.

Thus, if you have a large value of μ, you are unlikely to deviate from it by very much. If your variables have probability p of being equal to 1, then to get a probability of missing your expected value by ϵ to be less than $\frac{1}{N}$, you need a sample size of $s \geq \frac{2\ln(N)}{\epsilon^2 p}$. Because this required sample size is logarithmic in the total size of the problem, there is hope that the sampled problem will be tractable.

We now need to define what we mean by a dense graph.

Definition 1 *A graph G is δ-dense if every vertex has degree at least δN, where N is the number of vertices in the graph.*

We will develop, for any $\delta > 0$, a PTAS for δ-dense graphs; $\forall\delta > \epsilon, \forall\epsilon > 0$, this scheme will give us a $(1-\epsilon)$-approximation algorithm whose runtime is $N^{O(\frac{1}{\epsilon^2\delta})}$, which is impractical for small values of ϵ and δ.

1.3 Random sampling and 3-coloring δ-dense graphs

3-coloring dense graphs is a problem which we will use to demonstrate the techniques; given a graph $G = (V, E)$, color every vertex such that no edge has two endpoints of the same color. It is clear that if you know the colors of all neighbors of a particular vertex, you can pick a legal color for that vertex.

Sample k vertices at random, and guess their correct colors. Use these colors as hints to help you color the remainder of the graph. If you let k be sufficiently small, you can do a brute force search of all 3^k possibilities of the colorings of those vertices to find the correct one. Because the graph is dense, we can pick $k = \frac{2\ln(N)}{\delta}$ vertices, and is is probable that at least one neighbor of every vertex is chosen, and thus we have a hint for every vertex.

Claim: the probability that some vertex has no neighbor picked is small. The expected number of neighbors of a vertex v_i picked is $2\ln(N)$. Thus, by the Chernoff bound, the probability of having 0 neighbors picked is at most $e^{3ln(N)/2} = \frac{1}{N^{3/2}}$. thus, by the union bound, the probability that at least one vertex has no neighbors chosen is $N\frac{1}{N^{3/2}} = 1/\sqrt{N}$.

Since the sample is of size $O(\frac{\ln(N)}{\delta})$, you can exhaustively search all possible colorings of the sample in $O(N^{\frac{1}{\delta}})$ time.

Now we have a graph and every vertex has at least one colored neighbor. While some vertex has neighbors of two colors, the color of that vertex is obvious, so color it. Eventually, every vertex has exactly one neighboring color. Solving this problem is equivalent to solving a boolean constraint satisfaction problem with two variables per constraint (2-CSP). To see this, let c_i be the color of the colored neighbor of v_i. Then you have variables f_i which are true if color v_i will be assigned color $c_i + 1mod3$ and false if color v_i will be assigned color $c_i - 1mod3$. Each edge connecting two uncolored vertices i and j will create a constraint of the form $h(f_i, f_j)$ for some boolean function h, where the constraint is false precisely if the two endpoints of the edge receive the same color. 2-CSP is polytime-sovable if a solution exists, and we know that one does, so this colors the remainder of the graph in polynomial time.

1.4 Random sampling and MAX CUT in dense graphs

Switching back to MAX-CUT, we know that there is the local optimality condition that every vertex is opposite at least half of its neighbors. The idea of the next algorithm will be to sample the graph and use the sample to assign vertices, just as in the 3-coloring problem.

Once again, the sample size must be $O(\log(N))$, to keep complexity down. Thus, we sample $\frac{\log(N)}{\delta\epsilon^2}$ total vertices. Because the graph is dense, this gives you an expectation of getting $\frac{\log(N)}{\epsilon^2}$ neighbors of every vertex simultaneously, and by the Chernoff bound you get this many with high probability. Suppose 3/4 of the neighbors of a vertex are on one side of the cut. Then by the Chernoff bound, a sample of size $(\ln(N))/\epsilon^2$ will give us an accurate result with high probability. Unfortunately, this doesn't tell us what to do with those vertices where the fraction of edges on each side is about $\frac{1}{2}$, so we instead introduce an LP and use randomized rounding.

First, we introduce a quadratic integer program. Let $x_i = 0$ if vertex i will be assigned to the left set, 1 if it will be assigned to the right set. We want to maximize $\sum_{(i,j)\in E} x_i(1 - x_j)$. Because this is a quadratic objective function, it will be hard to solve, so rewrite it as $max \sum_i x_i \sum_{j:(i,j)\in E}(1 - x_j)$. The right-hand sum here is the number of left side neighbors of v_i; we will denote this value as $r_i(x)$. Now the objective is $max \sum_i x_i r_i(x)$. We have an estimate of r_i from the sample; call this estimate e_i. Now we have the linear objective $max \sum_i x_i e_i$.

We still need to ensure that our estimates were close to correct, so we want to add a new constraint that $r_i = e_i$. But, they are not expected to be exact, so instead we add the constraint $e_i - \epsilon N \leq r_i \leq e_i + \epsilon N$. We know that this integer program is solvable with high probability, because with high probability the estimates are close to correct. Since it is an integer program, however, we need to relax the 0-1 constraint so that it becomes a linear program.

Now, we need to show that there exists a feasible solution to the program, show that the optimum of the LP is near OPT, and show that we can round the LP to a cut usefully.

With high probability, e_i was within $\pm\epsilon$ of r_i for the optimal solution. Proof: we sampled $4\ln(N)\epsilon^2$ neighbors of vertex i; thus the probability that our estimate is off by more than ϵ is at most $1/N^2$ by the Chernoff bound, so the probability that any vertex has an estimate off by more than ϵ is at most $1/N$ by the union bound. This is sort of hand-waving the mean being large, but you can make it more rigorous by observing that since what we care about is the absolute, not the relative, deviation from the mean, if the mean is small we are deviating by a large relative factor, which is unlikely. Thus, the optimal solution is feasible, so there exists a feasible solution, and moreover, since the optimal solution is feasible, the solution to the LP has value at most OPT.

We will create the cut using a randomized rounding procedure; $\hat{x_j} = 1$ with probability equal to x_j and zero otherwise. $E(\hat{x_j}) = x_j$. By the Chernoff bound, we can show that $r_i(\hat{x}) \approx r_x(x)$, and $\sum_i \hat{x}_i r_i(\hat{x}) \approx \sum_i x_i r(x)$, so the expected value of our solution is approximately the value of the LP, so by the Chernoff bound, with high probability the value of our rounded solution is no more than $(1+\epsilon)$ times the LP value, which is less than opt, so the rounded solution is, with high probability, no more than $(1+\epsilon)OPT$.

This general technique can be applied to many problems which are "dense" enough that a sample provides a significant amount of information about the resulting problem, such as dense versions of MAX-2SAT and MAX-3SAT.