CS 810: Introduction to Complexity Theory 04/01/03

Lecture 19: Randomized Algorithms: MAXCUT

Instructor: Jin-Yi Cai Scribe: Christopher Hudzik, Sarah Knoop

1 Introduction

Let $G = (V, E)$ be a graph over n vertices. A cut C of G is a subset of E such that there exist $V_1, V_2 \subseteq V$ where V_1 and V_2 partition V, and for each $e \in C$, one of its vertices is in V_1 and the other is in V_2.

Definition 1 (MAXCUT). *A* MAXCUT *of a graph $G = (V, E)$ is a cut C such that $|C|$ is maximized over all cuts of G.*

Similarly to MAXCUT, MINCUT of G is defined as the minimum $|C|$ over all cuts of G. We know that MINCUT is in P. It can be solved using maximum network flow between all pairs of vertices. Also, note that MAXCUT of G is not the same as MINCUT of G^{C}, G's complement graph.

We know that MAXCUT is NP-complete, meaning that we do not know how to solve it efficiently. We do have approximations, however. To quantify the accuracy of our approximations we will introduce a new term. We want a polynomial-time algorithm that achieves a cut C such that

$$\frac{|C|}{|C^*|} \geq r$$

where C^* is a maximum cut. Such an algorithm is called an r-approximation.

2 Deterministic MAXCUT Approximation Algorithm

We can define a greedy algorithm that achieves a 1/2-approximation:
For a graph $G = (V, E)$ with $V = \{1, \ldots, n\}$, define $E_i = \{(k, i) \in E : k < i\}$. Initially, let $V_1 = \{1\}$ and $V_2 = \emptyset$. Then, for each i from 2 to n, add i to either V_1 or V_2 so that number of edges in E_i that are on the cut is maximized. We claim that this heuristic achieves 1/2-approximation.

Let C be the cut obtained by the algorithm. The disjoint sets $E_1, E_2, \ldots, E_n$ partition E. So, $|E| = \Sigma_i E_i$. For each $i \in V$, let $E'_i = E_i \cap C$. Then, $C = \bigcup_i E'_i$. As sets E_i are disjoint, the sets E'_i are also disjoint. Thus, $|C| = \Sigma_i |E'_i|$. The main observation is that for each $i \in V$, we have $E'_i \geq E_i/2$. We conclude that $|C| \geq |E|/2$. As the size of maximum cut $|C^*| \leq |E|$, $|C| \geq (1/2)|C^*|$.

3 Randomized MAXCUT Approximation Algorithm

We present a randomized 1/2-approximation algorithm for MAXCUT. Then we show that it can be derandomized in polynomial time. The main goal is to illustrate ideas of randomization and derandomization.

The randomized algorithm is very simple. Given a graph $G = (V, E)$, we assign each vertex independently with equal probability to either V_1 or V_2. This will give us a cut C of G, and we will show that the expected size of $C \geq (1/2)|C^*|$.

Consider an edge $(i, j) \in E$. $\Pr[(i, j) \in C] = 1/2$. For $e \in E$, define χ_e to be a random variable such that $\chi_e = 1$ if $e \in C$ and $\chi_e = 0$ if $e \notin C$. Then $|C| = \sum_{e \in E} \chi_e$. Thus,

$$\mathrm{E}[|C|] = \sum_{e \in E} \mathrm{E}[\chi_e] = \sum_{e \in E} \Pr[e \in C] = \frac{|E|}{2} \geq \frac{1}{2}|C^*|.$$

The first equality follows from linearity of expectation. For two random variables X and Y, $E[X + Y] = E[X] + E[Y]$. This formula holds even if X and Y are not independent.

4 Derandomization

The above algorithm chose random numbers from an exponential number of possibilities. Instead of choosing among exponentially many numbers, we give a randomized algorithm that chooses from polynomially many and show that the average cut size among the polynomially many is at least half the maximum cut size. Thus, to derandomize, we can look at all of these cuts and pick the largest. This will guarantee a cut of at least half the maximum size.

4.1 Universal Hash Functions

Definition 2 (Universal family of hash functions). *Let U and T be finite sets. Let S be an index set for a family of functions $\{h_s : U \to T\}_{s \in S}$. $\{h_s\}_{s \in S}$ is called a* universal family of hash functions *if $\forall \alpha, \beta \in T, \forall x, y \in U, x \neq y$,*

$$\Pr_{s \in S}[h_s(x) = \alpha \wedge h_s(y) = \beta] = \frac{1}{|T|^2}$$

Notice that the RHS of above equation $1/|T|^2$ is the probability of getting α and β when we choose two elements independently and uniformly at random from T.

For all x, the map $Z_x : s \mapsto h_s(x)$ is a random variable. Note that $Z_x(s) = h_s(x)$. We have $\forall x \neq y \in U$, $\forall \alpha, \beta \in T$, $\Pr_{s \in S}[Z_x(s) = \alpha \wedge Z_y(s) = \beta] = \frac{1}{|T|^2}$. Hence, $\forall \alpha \in T, \forall x, y \in U, x \neq y$,

$$\begin{aligned}
\Pr_{s \in S}[Z_x(s) = \alpha] &= \sum_{\beta \in T} \Pr_{s \in S}[Z_x(s) = \alpha \wedge Z_y(s) = \beta] \\
&= \sum_{\beta \in T} \frac{1}{|T|^2} \\
&= \frac{1}{|T|}
\end{aligned}$$

So, Z_x is a uniform random variable on T. From this, for any $x \neq y \in U$ and $\alpha, \beta \in T$,

$$\Pr_{s \in S}[Z_x(s) = \alpha \ \wedge \ Z_y(s) = \beta] = \Pr_{s \in S}[Z_x(s) = \alpha] \cdot \Pr_{s \in S}[Z_y(s) = \beta].$$

So, for any $x \neq y \in U$, the random variables Z_x and Z_y are independent. The set of random variables, $\{Z_x\}_{x \in U}$ are *pairwise independent.* The random variables in this set depend on each other, but if we pick any two of them, they will be independent.

We do not want to think in terms of the $Z_x(s)$. We will instead consider the $h_s(x)$.

Example:
Let p be a prime number. Then $\mathbb{Z}/p = \{0, 1, \ldots, p-1\}$ with the operations $+$ and $\cdot$ is a finite field. Consider the map $h_{s=(a,b)} : x \mapsto ax + b$ for $a, b \in \mathbb{Z}/p$. We will verify that $\{h_{(a,b)}\}_{a,b \in \mathbb{Z}/p}$ is a universal family of hash functions.

For all $x, y, \alpha, \beta \in \mathbb{Z}/p, x \neq y$, how many pairs $(a, b) \in \mathbb{Z}/p$ are there satisfying the equations

$$\begin{aligned} ax + b &= \alpha \\ ay + b &= \beta \ ? \end{aligned}$$

(In the above equations, a and b are the unknowns.) These equations are equivalent to $\begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$. The determinant of $\begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} = x - y \neq 0$. Therefore, there exists a unique solution such that this equation holds. Thus,

$$\Pr_{s=(a,b) \in (\mathbb{Z}/p)^2}[h_s(x) = \alpha \wedge h_s(y) = \beta] = \frac{1}{p^2}.$$

So, $\{h_{(a,b)}\}_{a,b \in \mathbb{Z}/p}$ is a universal family of hash functions.
⊠

This can be generalized to any finite field, namely the finite field of 2^k elements $GF(2^k)$, simply by a replacement of $\mathbb{Z}/p$.

4.2 MAXCUT Approximation Algorithm Using Universal Hash Functions

Let $G = (V, E)$ be a graph with $V = \{0, \ldots, n-1\}$. Set k so that $2^k \geq n > 2^{k-1}$. Choose a and b at random from $GF(2^k)$. Start with $V_1, V_2 = \phi$. For each $i \in V$, treat i as a member of $GF[2^k]$ compute $ai + b$. Assign i to either V_1 or V_2 according to the first bit of $ai + b$. We claim that the expected size of cut obtained is $\geq |E|/2$.

Let $\chi_{(a,b)}(i) =$ the first bit of $ai + b$. We know that $\{ai + b\}_{a,b \in GF(2^k)}$ is a universal family of hash functions. Thus, $\{\chi_{(a,b)}\}_{a,b \in GF(2^k)}$ is a universal family of hash functions. Then, a cut C obtained by the above randomized algorithm is given by $C = \{(i,j) | \chi_{(a,b)}(i) \neq \chi_{(a,b)}(j)\}$.

Because $\{\chi_{(a,b)}\}_{a,b \in GF(2^k)}$ is a universal family of hash functions, $\Pr[\chi_{(a,b)}(i) \neq \chi_{(a,b)}(j)] = 1/2$. Thus, using the analysis from Section 3, we have $\mathrm{E}[|C|] \geq |E|/2$.

We can derandomize the above algorithm in polynomial time. There are less than $4n^2$ different choices for (a, b). To derandomize, we can examine the cuts created by $\{\chi_{(a,b)}\}$ for all $a, b \in GF(2^k)$ in polynomial time. One of these cuts is guaranteed to be at least $|C^*|/2$ because $\mathrm{E}[|C|] \geq |C^*|/2$.

This gives us a deterministic r-approximation algorithm for MAXCUT.

This derandomized algorithm does not give a better approximation ratio than the greedy algorithm, but it is a parallel algorithm. For each pair (a, b), the determination of which side of the cut each vertex is on is independent of the other vertices . Thus, this can be executed in parallel. Additionally, the cut produced from each pair (a, b) is independent of the cuts from other pairs. Therefore, all cuts can be determined in parallel. After these cuts have been computed, the maximum can be found with an NC computation. Also, this algorithm is a good example of derandomization.