COMPSCI 638: Graph Algorithms November 8, 2019

Lecture 22

Lecturer: Debmalya Panigrahi *Scribe: Kevin Sun*

1 Overview

In this lecture, we prove that the maximum cut problem is NP-hard. We also present a few approximation algorithms, including a well-known result based on semidefinite programming.

2 Maximum Cut

The maximum cut problem is the maximization version of the minimum cut problem (see Lecture 8). We are given an undirected graph $G = (V, E)$ where each edge e has weight $w(e) \geq 0$. The goal is to find a cut $S \subset V$ that maximizes the total weight of cut edges. For simplicity, we assume $w(e) = 1$ for every $e \in E$, so our goal is to find a cut that maximizes the number of cut edges.

We first describe two simple 1/2-approximation algorithms. In particular, each of these algorithms will return at least $|E|/2$ edges, and since any cut contains at most $|E|$ edges, this implies that these algorithms return a cut of size at least half of the maximum cut size.

Local search: Start with any arbitrary cut, and while there exists any vertex u such that moving u to the other side of the cut would increase the cut value, make such a move. Let $(S, V \setminus S)$ denote the final cut, and let $d(u, S)$ denote the contribution of the degree of u to the cut value $|\delta(S)|$. Then

$$2 \cdot |\delta(S)| = \sum_{u \in S} d(u, S) + \sum_{v \in V \setminus S} d(v, S) \geq \sum_{u \in S} \frac{\deg(u)}{2} + \sum_{v \in V \setminus S} \frac{\deg(v)}{2} = \frac{1}{2} \sum_{x \in V} \deg(x) = |E|,$$

where the inequality holds due to the terminating condition of our algorithm.

Random placement: Place each vertex in S (initially empty) independently with probability 1/2. Then any edge $e = \{u, v\} \in E$ is cut if and only if exactly one of its endpoints is in S, and this occurs with probability 1/2. Thus, the expected contribution of e to $\delta(S)$ is 1/2, so by linearity of expectation, the expected size of $\delta(S)$ (summing over all edges) is $|E|/2$.

2.1 Max Cut is NP-Hard

We now prove that the maximum cut problem is NP-hard by showing a reduction from the maximum independent set (MIS) problem, which is known to be NP-hard. In MIS, we are given an undirected graph $G = (V, E)$. A subset of vertices S is an *independent set* if there does not exist any edge between two vertices. The goal is to find a independent set of maximum cardinality.

Theorem 1. *The maximum cut problem is NP-hard.*

Proof. Let $G = (V, E)$ be an instance of MIS. We construct a graph $G' = (V', E')$ as follows: first, add a vertex x to V and add an edge $\{x, u\}$ for every $u \in V$. Next, for each edge $e = \{u, v\} \in E$,

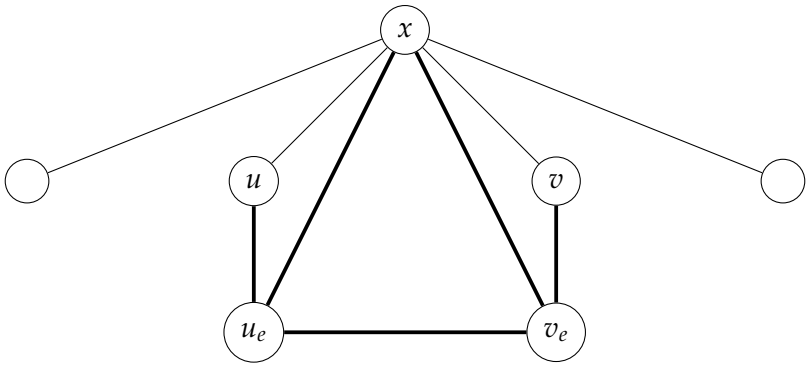

Figure 1: The graph G' constructed from the instance of MIS $G = (V, E)$, where in this case, $|V| = 4$ and there is one edge $\{u, v\} \in E$. The five edges of the gadget G_e are bold.

add two vertices u_e, v_e to V' and five edges $\{u_e, v_e\}, \{u, u_e\}, \{v, v_e\}, \{x, u_e\}, \{x, v_e\}$ to E' (see Fig. 1). These two vertices and five edges form the "gadget" G_e corresponding to edge $e \in E$.

We first prove that if G contains an independent set $I \subseteq V$ such that $|I| \geq k$, then there exists a cut S in G' such that $|\delta(S)| \geq k + 4 \cdot |E|$. Given such a set I, we construct S as follows: start with $S = I$. Then for every edge $e = \{u, v\} \in E$, do the following:

- If $u \in I$ and $v \notin I$, then add v_e to S. Similarly, if $u \notin S$ and $v \in S$, then add u_e to S. Notice that in either case, exactly four of the five edges of G_e are cut.
- If $u, v \notin I$, then add u_e and v_e to S. Again, notice that four of the five edges of G_e are cut.
- The case where $u, v \in I$ is not possible because I is an independent set while $\{u, v\}$ is an edge.

For every $w \in I$, the edge $\{x, w\}$ is cut by S. Furthermore, for every edge $e = \{u, v\} \in E$, regardless of the positioning of u and v relative to I, exactly four gadget edges of G_e are cut. Thus, the number of edges cut by S is $k + 4 \cdot |E|$, as desired.

Now we prove the converse: given a cut S in G' such that $|\delta(S)| \geq k + 4 \cdot |E|$, we shall construct an independent set $I \subseteq V$ such that $|I| \geq k$. Without loss of generality, we can assume $x \notin S$ because otherwise, we can swap S and $V \setminus S$ without affecting the cut value.

Let I be the subset of S corresponding to vertices in G, and suppose I is not an independent set. Consider any edge $e = \{u, v\} \in E$. If $u, v \in S$ (i.e., $\{u, v\} \in E$), then we can see that S cuts at most three edges of G_e. In all other cases (at least one of u, v is not in S), we saw a way for S to cut four gadget edges. Thus, letting $m(I)$ denote the number of edges within I, we have

$$|\delta(S)| \leq |I| + 3 \cdot m(I) + 4 \cdot (|E| - m(I)) = |I| + 4 \cdot |E| - m(I).$$

Since $|\delta(S)| \geq k + 4 \cdot |E|$, this implies $|I| \geq k + m(I)$. Thus, for each edge within I, we can "afford" to remove one of its endpoints from I to decrease the value of $m(I)$ by one. After doing this at most $m(I)$ times, we are left with an independent set of size at least k, as desired. □

2.2 Linear, Quadratic, and Semidefinite Programs

Our goal now is to give an α-approximation for the maximum cut problem for some $\alpha > 1/2$. (Recall that we gave two $1/2$-approximations at the beginning of this section). A simple observation

is the following: for any odd cycle, at least one edge cannot be a cut edge. This motivates the following LP formulation for our problem:

$$\begin{aligned}
\text{(LP-C): } \max & \sum_{(i,j)\in E} x_{ij} \\
\sum_{(i,j)\in C} x_{ij} &\leq |C| - 1 \quad \forall \text{ odd cycle } C \\
0 \leq x_{ij} &\leq 1 \quad \forall (i,j) \in E.
\end{aligned}$$

Unfortunately, there can be an exponentially large number of odd cycles in a graph, so we'd like to improve upon this formulation. It can be shown that the following is equivalent to (LP-C):

$$\begin{aligned}
\text{(LP-T): } \max & \sum_{(i,j)\in E} x_{ij} \\
\sum_{(i,j)\in T} x_{ij} &\leq 2 \quad \forall \text{ triangle } T \\
x_{ij} + x_{jk} &\geq x_{ik} \quad \forall i,j,k \in V \\
0 \leq x_{ij} &\leq 1 \quad \forall i,j \in V.
\end{aligned}$$

From this perspective, we see that our goal is to find a metric that satisfies the "triangle constraints" and maximizes the total distances across the set of edges. Of course, the benefit of this formulation is that it only has a polynomial number of constraints. And indeed, using this linear program it is possible to obtain an algorithm with approximation ratio better than $1/2$.

Unfortunately, it can be shown that the integrality gap of (LP-C), (LP-T), and any known LP formulation for the maximum cut problem, is two. Therefore, any LP-based algorithm is, at best, a $1/2$-approximation, but this is no better than the two simple algorithms presented at the beginning of this section.

Quadratic program: Since LP formulations do not seem to yield better algorithms, let us consider writing a quadratic program for the maximum cut problem. For any $S \subseteq V$ and $i \in V$, we represent $i \in S$ as $x_i = 1$, and $i \notin S$ as $x_i = -1$. Then an edge (i,j) is cut if and only if $(1 - x_i x_j)/2 = 1$, and otherwise, this quantity is zero. This leads us to the following quadratic program:

$$\begin{aligned}
\text{(QP): } \max & \sum_{(u,v)\in E} \frac{1 - x_i x_j}{2} \\
x_i &\in \{-1, 1\} \quad \forall u \in V.
\end{aligned}$$

Notice that the condition $x_i \in \{-1, 1\}$ is equivalent to allowing $x_i \in \mathbb{R}$ with the constraint $\|x_i\| = 1$. Our final formulation for maximum cut, a semidefinite program, will be constructed by increasing the dimension of the space in which the variable x_u resides.

Semidefinite program: We now consider extending the quadratic formulation into $n = |V|$ dimensions. Each scalar variable x_i is replaced by a vector $v_i \in \mathbb{R}^n$, and the constraint $\|x_i\| = 1$ becomes

$v_i^\top v_i = 1$. This gives us the following formulation:

$$\begin{aligned} \text{(SDP): } \max \sum_{(i,j)\in E} \frac{1 - v_i \cdot v_j}{2} \\ v_i^\top v_i = 1 \quad \forall i \in V \\ d_{ij} = v_i^\top v_j \quad \forall i, j \in V, \end{aligned}$$

where the last "constraint" is simply satisfied by setting $d_{ij} = v_i^\top v_j \in \mathbb{R}$. Notice that $v_i^\top v_i = 1$ is then equivalent to $d_{ii} = 1$, and the remaining constraints can be captured by $D = V^\top V$, where the i-th column of V is the unit vector v_i. Thus, the matrix D is positive semidefinite (see Lecture 17).

Recall that for D to be positive semidefinite, it is equivalent to require $x^\top D x \geq 0$ for every $x \in \mathbb{R}^n$. This is a set of (infinitely many) linear constraints, and separation oracles for this problem exist, so this semidefinite program can be solved in polynomial time. Once we have a solution (i.e., a set of n vectors in $\mathbb{R}^n$), we round it by setting S as the set of vertices whose corresponding vectors lie on one side of a hyperplane through the origin, chosen uniformly at random.

Theorem 2 (Goemans and Williamson [GW95])**.** *The randomized rounding algorithm described above is a* 0.878*-approximation for the maximum cut problem.*

Proof. Let (i, j) be an arbitrary edge of the graph, θ_{ij} be the angle formed by the vectors corresponding to vertices i and j, and S denote the output solution. The probability that S cuts edge (i, j) is θ_{ij}/π so by linearity of expectation,

$$\mathbb{E}\left[|\delta(S)|\right] = \sum_{(i,j)\in E} \Pr((i,j) \text{ is cut by } S) = \sum_{(i,j)\in E} \frac{\theta_{ij}}{\pi}$$

On the other hand, the objective value obtained by solving (SDP) is

$$\sum_{(i,j)\in E} \frac{1 - v_i \cdot v_j}{2} = \sum_{(i,j)\in E} \frac{1 - \cos\theta_{ij}}{2}.$$

Thus, the approximation ratio of this algorithm can be computed as follows:

$$\min_{\theta_{ij}} \frac{\theta_{ij}/\pi}{(1 - \cos\theta_{ij})/2} \approx 0.878. \qquad \square$$

Remark: The analysis of Theorem 2 seems somewhat slack. If θ denotes the argument found in the computation of the approximation ratio, it seems that in order for the analysis to be tight, every angle formed by the vectors must have value roughly θ. Yet, assuming a well-known conjecture known as the Unique Games Conjecture, this approximation ratio of roughly 0.878 is optimal, i.e., it is NP-hard to approximate the maximum cut problem to any smaller factor.

3 Summary

In this lecture, we gave two simple 1/2-approximation algorithms for the maximum cut problem. We then showed that it is NP-hard via a reduction from the maximum independent set problem. Finally, we presented an algorithm based on semidefinite programming due to Goemans and Williamson [GW95] that has an approximation ratio of roughly 0.878.

References

[GW95] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.