COMPSCI 632: Approximation Algorithms October 25, 2017

Lecture 16: Maximum Cut, Integerality Gap and Metric Embedding

Lecturer: Debmalya Panigrahi *Scribe: Fred Zhang*

1 Overview

We discuss the Goemans & Williamson's algorithm for graph maximum cut [GW95]. We also show tight integrality gaps for the linear programming relaxations for the vertex cover and set cover problem. Finally, we introduce the randomized low-distortion embedding from a graph metric to a tree metric [Bar96, FRT04].

2 Maximum Cut

In the last lecture, we introduced the classic problem of maximum cut, where the objective is to find a cut of the given graph with maximum weight. We also saw some techniques give $1/2$-approximation, such as random assignment and local search. In this lecture we will introduce the semidefinite-programming (SDP) relaxation for this problem. This approach, proposed by Goemans & Williamson in 90's, improves upon $1/2$ and is in fact the best we can do, under certain complexity theory hypothesis.

2.1 Quadratic Integer Program

To begin with, let's first formulate the maximum cut problem an integer problem and then consider a relaxation. We represent the cut as a partition of the vertices and define the variables

$$x_u = \begin{cases} 1 & \text{if } u \in S \\ -1 & \text{if } u \in V \setminus S \end{cases}.$$

Now the maximum cut problem can be formulated as

$$\begin{aligned} \text{maximize} \quad & \frac{1}{2} \sum_{(i,j)\in E} \left(1 - x_i \cdot x_j\right) \\ \text{such that} \quad & x_i \in \{-1, +1\}. \end{aligned} \tag{integrality}$$

Observe that the objective is precisely the value of the cut. If $x_i \neq x_j$, *i.e.*, they are not on the same side, then edge (i, j) contributes 1 to the objective. Otherwise, it contributes 0. Now let us slightly rewrite this program by defining $y_{ij} = x_i x_j$.

$$\begin{aligned} \text{maximize} \quad & \frac{1}{2} \sum_{(i,j)\in E} \left(1 - y_{ij}\right) \\ \text{such that} \quad & y_{ii} = 1 \\ & \boldsymbol{Y} = \boldsymbol{x}\boldsymbol{x}^T. \end{aligned} \tag{rank-1}$$

The last constraint says that matrix $\boldsymbol{Y}$ can be expressed as an outer product of a vector $\boldsymbol{x}$ with itself. Combined with the first constraint, $Y_{ij} = x_j x_j = 1$, this implies integrality: $x_i = \{-1, +1\}$. On the other hand, given

a solution $\boldsymbol{x}$ to the first integer program, trivially $\boldsymbol{Y} = \boldsymbol{x}\boldsymbol{x}^T$ is a solution to the second. Thus, these two formulations are exactly equivalent.

At this moment, solving them would give an optimal integral solution, so they are still NP-hard to solve. Notice that the last constraint (rank-1) imposes that $\boldsymbol{Y}$ is of rank 1. Relaxing it gives a semidefinite program.

2.2 Semidefinite Programming Relaxation

To arrive at an SDP relaxation, we replace the rank-1 constraint by a positive semidefinite constraint.

$$\begin{aligned} \text{maximize} \quad & \frac{1}{2} \sum_{(i,j)\in E} \left(1 - y_{ij}\right) \\ \text{such that} \quad & y_{ii} = 1 \\ & \boldsymbol{Y} \succeq 0. \end{aligned} \tag{PSD}$$

Recall that we use $\boldsymbol{Y} \succeq 0$ to denote that matrix $\boldsymbol{Y}$ is positive semidefinite (PSD). Now is this relaxation polynomial-time solvable? In fact, SDP is a tractable problem. Observe that the PSD constraint is an *almost* linear constraint. As we introduced last time, it is equivalent of

$$\forall \boldsymbol{z} \in \mathbb{R}^n : \quad \boldsymbol{z}^T \boldsymbol{Y} \boldsymbol{z} \geq 0,$$

This is not a quadratic constraint, since $\boldsymbol{z}$ is a constant. But we simply have a linear constraint for every such $\boldsymbol{z}$, and of course, there are infinitely many of them. It turns out that this constraint has a efficient separation oracle (*hint*: consider the eigendecomposition of $\boldsymbol{Y}$), so the Ellipsoid method can give an optimal solution in polynomial time.

Next, we rewrite the SDP to make it more explicit. Again, we claimed last lecture that the PSD constraint is also equivalent of

$$\text{there exists } \boldsymbol{V} \text{ such that } \boldsymbol{Y} = \boldsymbol{V}\boldsymbol{V}^T.$$

In other words, each entry Y_{ij} can be expressed as the inner product of two vectors.

$$\begin{aligned} \text{maximize} \quad & \frac{1}{2} \sum_{(i,j)\in E} \left(1 - \boldsymbol{v}_i \boldsymbol{v}_j\right) \\ \text{such that} \quad & \|\boldsymbol{v}_i\|_2 = 1 \\ & \boldsymbol{v}_i \in \mathbb{R}^n \end{aligned}$$

The SDP relaxation embeds each vertex as a point on the n-dimensional unit sphere. Further, the objective would tend to put points of the same edge at diametrically opposed positions. This is because when $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are antipodal, $\boldsymbol{v}_i \cdot \boldsymbol{v}_j = -1$, and its contribution to the objective is maximized.

2.3 Algorithm and Analysis

Algorithm. Now with an optimal solution, how do we round it? We use a simple method: take a random hyperplane going through the origin, this cuts the unit sphere into two halves, naturally inducing a cut on the original graph. Intuitively, the algorithm makes sense because vertices of the same edge are far on the unit sphere, and a uniform random hyperplane would probably separate them.

Analysis. To formally analyze the algorithm, we notice that since $\boldsymbol{v}_i$ is of length 1, $\boldsymbol{v}_i \cdot \boldsymbol{v}_j$ is simply $\cos\theta_{ij}$, where θ_{ij} is the angle between $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$. Now a random hyperplane, projected onto this circle, is

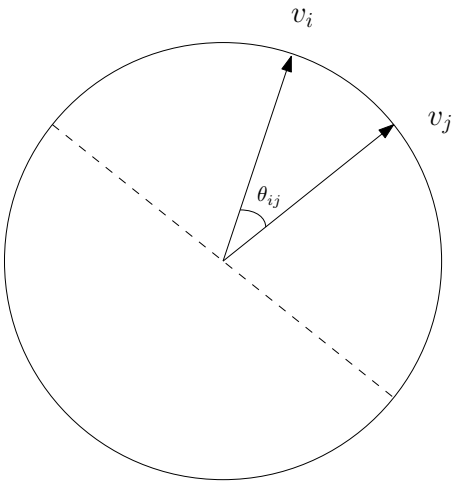

Figure 1: Picture for $\boldsymbol{v}_i, \boldsymbol{v}_j$, and the dashed line indicates a random hyperplane.

simply uniform random number in $[0, \pi]$, and it lands within the angle between $\boldsymbol{v}_i, \boldsymbol{v}_j$ with probability $\frac{\theta}{\pi}$. Therefore, the probability that a random hyperplane separates the $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ is $\frac{\theta_{ij}}{\pi} = \frac{\arccos \boldsymbol{v}_i \cdot \boldsymbol{v}_j}{\pi}$. Using the linearity of expectations and also incorporating weights, we have

$$\begin{aligned}\mathbb{E}[\text{value of the output cut}] &= \sum_{(i,j)\in E} w_{ij} \cdot \Pr[(i,j) \text{ in the cut}] \\ &= \sum_{(i,j)\in E} w_{ij} \cdot \frac{\theta_{ij}}{\pi}.\end{aligned}$$

But at this point, how do we relate this to the SDP solution value? Note that for each $(i, j) \in E$, it contributes $\frac{1}{2}\left(1 - \theta_{ij}\right)$ to the SDP value but $\frac{\theta_{ij}}{\pi}$ to the algorithm's solution (modulo the weight). In fact, we can get a worst-case ratio between this two quantities by evaluating

$$\min_{0\le\theta\le\pi} \frac{\theta/\pi}{\frac{1}{2}\left(1 - \theta_{ij}\right)}$$

By simple calculus one can show that this is roughly 0.878. Therefore, the algorithm approximates maximum cut by a factor of 0.878 in expectation. We remark that the algorithm can be easility derandomized via conditional expectations [MR99].

This algorithm is knwon to be tight. The SDP relaxation indeed has a integrality gap that matches exactly this number. Also, more generally, no polynomial-time algorithm can beat 0.878, assuming the Unique Game Conjecture (UGC). Of course, there is no consensus about UGC at this moment, unlike $P \neq NP$, and this barrier might be overcome in the futre.

3 Integrality Gaps

This brings us to the second topic of today on constructing integrality gap instances. Let us start with a simple problem, the vertex cover.

3.1 Vertex Cover

Recall the following linear programming relaxation for the minimum vertex cover problem.

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e x_e & \\ \text{such that} \quad & x_u + x_v \geq 1 & \forall (u,v) \in E \\ & x_u \geq 0 & \forall u \in V. \end{aligned}$$

Also recall that we obtained a deterministic rounding algorithm, based on this LP, that achieves 2-approximation. Can we do better than 2 using this LP? To construct a nearly matching integrality gap, consider the complete graph:

- The integral solution needs to select $n-1$ vertices to cover all edges;
- The fractional solution can put $x_u = 1/2$ for every vertex u, covering all edges and giving a total cost of $n/2$.

Hence, the integrality gap is at least $2\left(1 - \frac{1}{n}\right)$. We cannot do much better than 2 with this LP.

3.2 Set Cover

Now let's switch to a more involved example, minimum set cover. Here, we given a set E of element and a collection $\mathcal{S}$ of subsets of E.

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathcal{S}} c_S x_S & \\ \text{such that} \quad & \sum_{S : e \in S} x_S \geq 1 & \forall e \in E \\ & x_S \geq 0 & \forall S \in \mathcal{S}. \end{aligned}$$

Let us construct an instance, a set system, that gives the desired integrality gap $\Theta(\log |E|)$. Consider the set of all binary vectors of lenght l with equal number of 1's and 0's, and there are $t = 2^l - 1$ many of them. They look like

$$\begin{aligned} \boldsymbol{v}_1 &= (1,0,1,0,1,0,1,0\cdots) \\ \boldsymbol{v}_2 &= (1,1,0,0,1,1,0,0\cdots) \\ \boldsymbol{v}_3 &= (1,1,1,0,0,0,1,1\cdots) \\ \boldsymbol{v}_4 &= (1,0,0,1,1,0,0,1\cdots), \end{aligned}$$

and so on. Let $E = [t]$ be the ground set. To construct a set system , let $v_i(j)$ denote the jth entry of $\boldsymbol{v}_i$ and

$$S_i = \{j : v_j(i) = 1\}.$$

Here, we are taking the ith column from this table of vectors $\boldsymbol{v}_i$ and think of it as a characterstic vector of S_i. With this construction, let us now reason about the integrality gap.

- Any integral solution has to pick at least $l/2 + 1$ sets. Suppose it picks $l/2$ sets. Consider the element i with $v_i(j) = 0$ for all j such that S_j is chosen and $v_i(j) = 1$ otherwise. Clearly, i is not covered.
- For the fractional solution, each element appears in half of the sets. By symmetry, setting $x_S = 2/l$ for all S covers all elements, for a total cost of 2.

Therefore, the set cover problem has an integrality gap of $\Omega(\log |E|)$.

4 Distance Preserving Embedding

Let us move onto our next topic, called *distance preserving embeddings* or *low-distortion embeddings*. The motivation is that sometimes the problem comes with certain metric, for example, a graph metric, that tends to be hard to work with. We would like to embed it into a simple metric, for example, a tree metric, where the distance is defined simply as length of the path between two vertices. We want that the embedding does not distort the original metric space very much so that a good solution on the simple space approximates the problem well.

4.1 Tree Metric Embedding

Here is a result due to [Bar96] and [FRT04]. They achieve the optimal construction for embedding a graph metric into a tree metric.

Theorem 1. *Given a graph $G = (V, E)$, there exists a distribution $\mathcal{D}$ on trees on V such that*

(1) $\mathbb{E}_{T\sim\mathcal{D}}[d_T(u,v)] \leq O(\log n) \cdot d_G(u,v)$ *for all* $u, v \in V$.

(2) $d_T(u,v) \geq d_G(u,v)$ *for all* $T \in \mathcal{D}$ *and* $u, v \in V$,

where $d_T(\cdot,\cdot)$ and $d_G(\cdot,\cdot)$ denote the tree and graph metric, respectively. Further, there exists polynomial time algorithm that outputs a sample from this distribution.

The first condition claims that each pairwise distance is not stretched very much expectation. The second claims that no tree in this distribution shrinks any distance. Let us see why randomization is necessary for the first condition. Consider a cycle. The second condition forbids adding any chord, as it would decrease the distance between the its endpoints. However, removing any edge would increase the distance of the edge's two endpoints by a factor of n. Hence, one cannot hope this may hold true deterministically. Randomization saves us in this scenario. We could delete a uniformly random edge, and in expectation every distance is distorted roughly by a factor of 2.

4.2 Group Steiner Tree

Next, we apply this tree embedding technique on the problem of *group Steiner tree*. It is a natural generalization of the Steiner tree problem. Given groups of terminals $\{g_i\}_{i=1}^m$ with $g_i \subseteq V$, it is required to find a minimum weight tree that contains at least one terminal from every group. We do not know any direct algorithm on this problem. All algorithms rely on the tree embeddings, so let us also focus on the case when the input graph G is a tree.

References

[Bar96] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of 37th Annual Symposium on Foundations of Computer Science, 1996*, pages 184–193, 1996.

[FRT04] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485 – 497, 2004. Special Issue on STOC 2003.

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

[MR99] Sanjeev Mahajan and Hariharan Ramesh. Derandomizing approximation algorithms based on semidefinite programming. *SIAM Journal on Computing*, 28(5):1641–1663, 1999.