EECS598-001
Approximation Algorithms & Hardness of Approximation

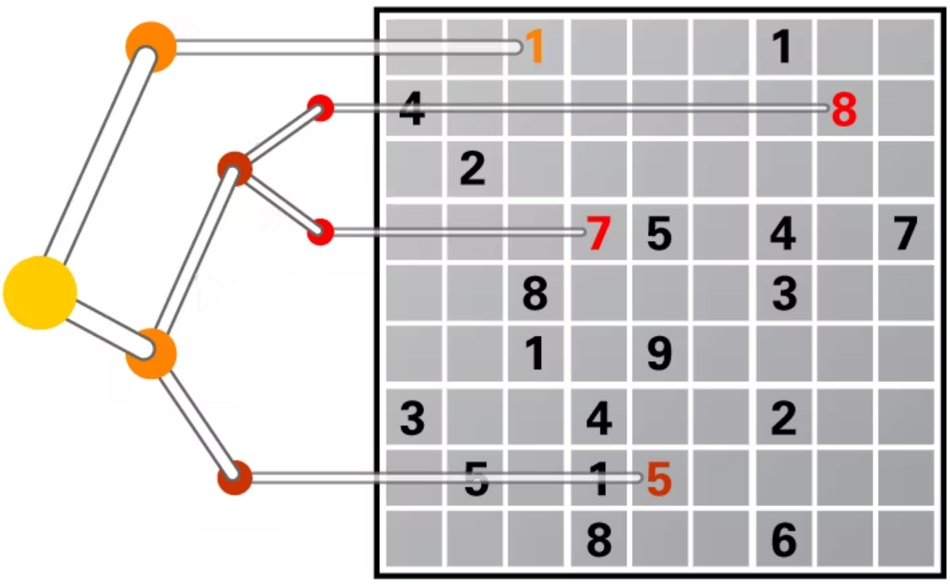
Pingbang Hu

March 27, 2024

Abstract

This is an advanced graduate-level algorithm course taught by Euiwoong Lee at University of Michigan. Topics include both approximation algorithms like covering, clustering, network design, and constraint satisfaction problems (the first half), and also the hardness of approximation algorithms (the second half).

The first half of the course is classical and well-studied, and we'll use Williamson and Shmoys [WS11], Vazirani [Vaz02] as our reference. The second half of the course is still developing, and we'll look into papers by Barak and Steurer [BS14], O'donnell [ODo21], etc.



This course is taken in Fall 2022, and the date on the cover page is the last updated time.

Contents

Chapter 1

Introduction

Lecture 1: Overview and Set Cover

1.1 Computational Problem

29 Aug. 10:30

We're interested in the following optimization problem: Given a problem with an input, we want to either maximize or minimize some objectives. This suggests the following definition.

Definition 1.1.1 (Computational problem). A *computational problem* P is a function from input I to (X, f), where X is the feasible set of I and f is the objective function.

We see that by replacing f with $-f$, we can unify the notion and only consider either minimization or maximization, but we will not bother to do this.

Example (s-t shortest path). The s-t *shortest path* problem P can be formalized as follows. Given input I, it defines

- Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and two vertices $s, t \in \mathcal{V}$.
- Feasible set: $X = \{\text{set of all (simple) paths } s \text{ to } t\}$.
- objective function: $f\colon X \to \mathbb{R}$ where $f(x) = \text{length}(x)$ ($\#$ of edges of x).

The output of P should be some $x \in X$ (i.e., some valid s-t paths) such that it minimizes $f(x)$.

We see that the computational problem we focus on is an optimization problem, and more specifically, we're interested in combinatorial optimization.

Definition 1.1.2 (Combinatorial optimization). A *combinatorial optimization* problem is a problem where the feasible set X is a finite set.

Example (s-t shortest path). The s-t shortest path problem is a combinatorial optimization problem since given a graph $\mathcal{G}$ with $n = |\mathcal{V}|$, $m = |\mathcal{E}|$, there are at most $n!$ different paths, i.e., $|X| \leq n! < \infty$.

Note. We'll also look into some continuous optimization problem, where X is now infinite (or even uncountable). For example, find $x \in \mathbb{R}$ that minimizes $f(x) = x^2 + 2x + 1$. In this case, $X = \mathbb{R}$ which is uncountable (hence infinite).

1.2 Efficient Algorithms

Given a problem P, we want to solve it fast with algorithms. Before we characterize the speed of an algorithm, we should first define what exactly an algorithm is.

Definition 1.2.1 (Algorithm). Given a problem P and input I (which defines X and f), an *algorithm* A outputs solution $y = A(I)$ such that $y \in X$ and $y = \arg\max_{x \in X} f(x)$ or $\arg\min_{x \in X}$, depending on I.

Definition 1.2.2 (Efficient). We say that an algorithm A is *efficient* if it runs in **polynomial time**.

Remark (Runtime parametrization). The *runtime* of an algorithm A should be parametrized by the size of input I. Formally, given input I represented in s bits, runtime of A on I should be $\mathsf{poly}(s)$ for A to be efficient.

Note. In most cases, there are 1 or 2 parameters that essentially define the size of input.

Example (Graph). A natural representation of a graph with n vertices and m edges are

(a) Adjacency matrix: n^2 numbers.

(b) Adjacency list: $O(m + n)$ numbers.

Example (Set system). A set system with n elements and m sets has a natural representation which uses $O(nm)$ numbers.

Example. If an input I can be represented by s bits, then the runtime of an algorithm can be $O(s \log s)$, $O(s^2)$, or $O(s^{100})$, which are considered as efficient. On the other hand, something like 2^s or $s!$ are not.

Hence, our goal is to get $\mathsf{poly}((n, m))$-time algorithm!

1.3 Approximation Algorithms

We first note that many interesting combinatorial optimization problems are NP-hard, hence it's impossible to find optimum in polynomial time unless P is NP. This suggests one problem: *How well can we do in polynomial time?*

In normal cases, we may assume that objective function value is always positive, i.e., $f\colon X \to \mathbb{R}^* \cup \{0\}$. Then, we have the following definition which characterize the *slackness*.

Definition 1.3.1 (Approximation algorithm). Given an algorithm A, we say A is an *α-approximation algorithm* for a problem P if for every input I of P,

- Min: $f(A(I)) \leq \alpha \cdot \mathsf{OPT}(I)$ for $\alpha \geq 1$
- Max: $f(A(I)) \geq \alpha \cdot \mathsf{OPT}(I)$ for $\alpha \leq 1$

where we define $\mathsf{OPT}(I)$ as $\max_{x \in X} f(x)$ for maximization, $\min_{x \in X} f(x)$ if minimization.

We see that α characterizes the slackness allowed for our algorithm A. Now, we're ready to look at some interesting problems. Broadly, there are around 10 classes of them which are actively studied:

- We'll see cover, clustering, network design, and constraint satisfaction problems.
- We'll not see: graph cuts, Packing, Scheduling, String, etc.

The above list is growing! For example, applications of continuous optimization in combinatorial optimization is getting attention recently. Also, there are around 8 techniques developed, e.g., greedy, local search, LP rounding, SDP rounding, primal-dual, cuts and metrics, etc.

1.4 Hardness

For most problems we saw, we can even say that getting an α-approximation is NP-hard for some $\alpha > 1$. This bound is sometimes tight, but not always, and we'll focus on this part in the second half of this course.

Chapter 2

Covering

2.1 Set Cover

Before we jump into any problem formulations, we define a fundamental object in combinatorial optimization, the set system.

Definition 2.1.1 (Set system)**.** Given a ground set Ω (often called *universe*), the *set system* is an order pair $(\Omega, \mathcal{S})$ where $\mathcal{S}$ is a collection of subsets of Ω.

Note. For a set system $(\Omega, \mathcal{S})$, we often let $m := |\mathcal{S}|$ and $n := |\Omega|$.

Definition 2.1.2 (Degree)**.** Given a set system $(\Omega, \mathcal{S})$, the *degree of* $x \in \Omega$, $\deg(x)$, is defined as

$$\deg(x) := |\{S \in \mathcal{S} \mid x \in S\}| .$$

Remark (Bipartite representation)**.** Naturally, for a set system, we have a bipartite representation.

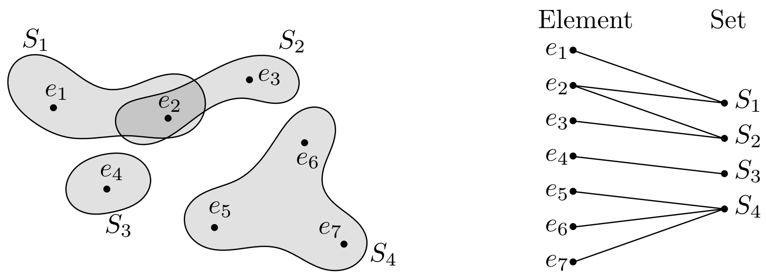


Figure 2.1: Bipartite representation of a set system.

Denote $d := \max_{e \in U} \deg(e) \leq m$ and $k := \max_{i \in [m]} |S_i| \leq n$, which is just the maximum vertex degree on two sides of the bipartite graph representation of this set system.

Finally, we have the following.

Definition 2.1.3 (Covering)**.** A *covering* $\mathcal{S}' \subseteq \mathcal{S}$ of $(\Omega, \mathcal{S})$ is a (sub)collection of subsets such that $\bigcup_{S \in \mathcal{S}'} S = \Omega$.

Let's first consider the classical problem called set cover.

Problem 2.1.1 (Set cover)**.** Given a finite set system $(U, \mathcal{S})$ where $\mathcal{S} := \{S_i \subseteq U\}_{i=1}^{m}$ along with a weight function $w \colon \mathcal{S} \to \mathbb{R}^+$, find a covering $\mathcal{S}'$ while minimizing $\sum_{S \in \mathcal{S}'} w(S)$.

Assuming there always exists at least one covering, we can in fact get two types of non-comparable approximation ratio in terms of k and d. Specifically, we get $\log k$ and d-approximation ratio via either greedy, LP rounding or dual-methods.

2.2 Greedy Method

We first see the algorithm when $w(S) = 1$ for all $S \in \mathcal{S}$.

Algorithm 2.1: Set cover – Greedy

Data: A set system $(U, \mathcal{S})$
Result: A covering $\mathcal{S}'$

```
1 S' ← ∅, i ← 0
2 while U ≠ ∅ do                                      // O(n)
3     Choose S_i with maximum |U ∩ S_i|                // O(mn)
4     for e ∈ U ∩ S_i do
5         y_e ← w(S_i)/|U ∩ S_i|                       // Average costs
6     S' ← S' ∪ {S_i}
7     U ← U \ S_i
8     i ← i + 1
9 return S'
```

We focus on the case that $w(S) = 1$ for all S.

Remark. It's clear that Algorithm 2.1 is a polynomial time algorithm, also, the output $\mathcal{S}'$ is always a valid covering.

Theorem 2.2.1. Algorithm 2.1 is an H_k-approximation[a] algorithm.

[a] H_k is the so-called *harmonic number*, which is defined as $\sum_{i=1}^{k} 1/i \leq \ln k + 1$.

Proof. Denote the OPT as $\mathcal{S}^* := \{S_1^*, \dots, S_\ell^*\}$, and first note that the average cost y_e essentially maintains $\sum_{e \in U} y_e = |\mathcal{S}'|$, hence we just need to bound y_e w.r.t. S^*. To do this, for any $S^* \in \mathcal{S}^*$, say $S_1^* =: \{e_1, \dots, e_k\}$ where we number e_i in terms of the order of which being deleted, i.e., e_1 is deleted first from U (line 7), etc.

Note. S_1^* can have less than k element, but in that case similar argument will follow. Also, if some elements are deleted at the same time, we just order them arbitrarily.

Then, we have the following claim.

Claim. For all e_i, $y_{e_i} \leq \frac{1}{k-i+1}$.

Proof. Consider the iteration when e_i was picked by S', i.e., $|U \cap S'| \geq |U \cap S_1^*| \geq k - i + 1$, then by definition (line 7) we have $y_{e_i} = \frac{1}{|U \cap S'|} \leq \frac{1}{|U \cap S_1^*|} \leq \frac{1}{k-i+1}$. ⊛

We immediately see that whenever the optimal solution pays 1 (for choosing S_1^* for instance), Algorithm 2.1 pays at most H_k since $\sum_{e_i \in S_1^*} y_{e_i} \leq \sum_{i=1}^{k} \frac{1}{k-i+1} = H_k$, or more formally,

$$|\mathcal{S}'| = \sum_{e \in U} y_e \leq \sum_{S_i^* \in \mathcal{S}^*} \underbrace{\sum_{e \in S_i^*} y_e}_{\leq H_k} \leq \ell \cdot H_k = H_k \cdot |\mathsf{OPT}|,$$

which finishes the proof. ■

In all, observe that $H_k \leq \ln k + 1$, we see that Algorithm 2.1 is a $(\ln k)$-approximation algorithm. Also, the weighted version can be easily derived by replacing 1 with the corresponding weight.

Lecture 2: Linear Programming with Set Covers

2.3 Linear Programming Rounding

31 Aug. 10:30

To get a d-approximation algorithm, instead of seeing the greedy algorithm, we first see the LP[1] dual method, which turns out to be exactly the same as the greedy algorithm.

> **As previously seen.** Both linear programming and convex programming can be solved in polynomial time.

Notice that it's more natural to define set cover in terms of ILP (integer LP). Define our integer variables $\{x_i\}_{i\in[n]}$ such that

$$x_i = \begin{cases} 1, & \text{if } S_i \in \mathcal{S}'; \\ 0, & \text{otherwise.} \end{cases}$$

In this way, we have the following ILP formulation for set cover as

$$\begin{aligned} \min \ & \sum_i w_i \cdot x_i \\ & \sum_{S_i \ni e} x_i \geq 1 \quad \forall i \in U \\ (\mathrm{IP}) \quad & x_i \in \{0,1\} \qquad \forall i. \end{aligned}$$

But we know that this is a NP-hard problem, so we relax it to be

$$\begin{aligned} \min \ & \sum_i w_i \cdot x_i \\ & \sum_{S_i \ni e} x_i \geq 1 \quad \forall i \in U \\ (\mathrm{LP}) \quad & x_i \geq 0 \qquad\qquad \forall i. \end{aligned}$$

Write it in a more compact form, we have

$$\begin{aligned} \min \ & \langle w, x\rangle \\ & Ax \geq \mathbb{1} \\ & x \geq 0 \end{aligned}$$

where $A \in \mathbb{R}^{n\times m}$ such that

$$A_{ij} = \begin{cases} 1, & \text{if } e_i \in S_j; \\ 0, & \text{otherwise.} \end{cases}$$

> **Note.** Note when we do relaxation, we want $x \in fes(\mathrm{IP}) \Rightarrow x \in \mathrm{fes}(\mathrm{LP})$, i.e., $fes(\mathrm{LP}) \supseteq fes(\mathrm{IP})$. Note that in this case, for a minimization problem, we have
>
> $$f(x) = \mathsf{OPT}_{\mathrm{LP}} \leq \mathsf{OPT}_{\mathrm{IP}}.$$

In this case, we see that the most natural way to get an integer solution from the fractional solution obtained from the relaxed LP is to **round** x to integral solution. This leads to the following algorithm.

Algorithm 2.2: Set cover – LP Rounding

Data: A set system $(U, \mathcal{S})$
Result: A covering $\mathcal{S}'$

```
1 x ← solve(LP)                                  // Solve the relaxed (LP)
2 S' ← {S_i : x_i ≥ 1/d}
3 return S'
```

We now prove the correctness and Algorithm 2.2's approximation ratio.

[1] See MATH561 for a complete reference.

Lemma 2.3.1. $\mathcal{S}'$ is a covering.

Proof. Fix $e \in U$, let $S_1, \dots, S_d$ be the sets containing e. We see that

$$\sum_{i=1}^{d} x_i \geq 1 \Rightarrow \exists j \in [d] \text{ s.t. } x_j \geq \frac{1}{d} \Rightarrow S_j \in \mathcal{S}'.$$

■

Theorem 2.3.1. Algorithm 2.2 is d-approximation algorithm.

Proof. By comparing $w(\mathcal{S}')$ and $\mathsf{OPT}_{\text{LP}} = \sum_{i=1}^{m} x_i w_i$, we see that

$$\mathsf{OPT} \leq \sum_{S_i \in \mathcal{S}'} w_i \leq d \sum_{S_i \in \mathcal{S}'} w_i x_i \leq d \cdot \mathsf{OPT}_{\text{LP}} \leq d \cdot \mathsf{OPT},$$

which implies $\mathsf{OPT}/d \leq \mathsf{OPT}_{\text{LP}} \leq \mathsf{OPT}$.

> **Note.** Note that OPT is assumed to be OPT_{IP}, i.e., the optimum of the original IP formulation of Problem 2.1.1.

■

Definition 2.3.1 (Intgrality gap). Given an integer programming, the *integrality gap* between OPT and OPT_{LP} of its LP relaxation is defined as

$$\sup_{\text{input } I} \frac{\mathsf{OPT}(I)}{\mathsf{OPT}_{\text{LP}}(I)}.$$

> **Remark.** We see that the integrality gap of Algorithm 2.2 is d from Theorem 2.3.1.

2.3.1 Randomized Linear Programming Rounding

And indeed, we can use a more natural way to do the rounding, i.e., respect to the x_i value.

> **Intuition.** If x_i is close to 1, it's reasonable to include it, vice versa.

We see that algorithm first.

Algorithm 2.3: Set cover – Randomized LP Rounding

Data: A set system $(U, \mathcal{S})$
Result: A (possible) covering $\mathcal{S}'$

```
x ← solve(LP)                         // Solve the relaxed (LP)
S ← ∅
for i = 1, ..., m do
    add S_i to S' w.p. x_i            // independently
return S'
```

Now, the question is, how is this $\mathcal{S}'$'s quality? In other words, fix $e \in U$, what's $\Pr(e \text{ is covered})$?

Lemma 2.3.2. $\Pr(e \text{ is covered}) \geq 1 - 1/e \approx 0.63$.

Proof. We bound $\Pr\left(\overline{e \text{ is covered}}\right)$ instead. Say $S_1, \dots, S_d$ are the sets containing e, then we see that

$$\Pr\left(\overline{e \text{ is covered}}\right) = \prod_{i=1}^{d}(1-x_i) \leq \prod_{i=1}^{d} e^{-x_i} = e^{-\overbrace{(x_1+\dots x_d)}^{\geq 1}} \leq e^{-1}.$$

Note. For every x, we have $1+x \leq e^x$, and this approximation is close when $|x|$ is small.

■

A standard way to boost the correctness of a randomized algorithm is to run it multiple time, which leads to the following.

Algorithm 2.4: Set cover – Multi-time Randomized LP Rounding

Data: A set system $(U, \mathcal{S})$, α
Result: A (possible) covering $\mathcal{S}'$

```
x ← solve(LP)                                   // Solve the relaxed (LP)
S ← ∅
for t = 1, ..., α do                            // independently
    for i = 1, ..., m do
        add S_i to S' w.p. x_i                  // independently
return S'
```

Lemma 2.3.3. With $\alpha = 2\ln n$, $\mathcal{S}'$ returned from Algorithm 2.4 is a covering w.p. at least $1 - \frac{1}{n}$.

Proof. We have $\Pr(e \text{ is not covered}) \leq e^{-\alpha}$ from independence of each run. Let $\alpha = 2\ln n$, then $\Pr(e \text{ is not covered}) \leq e^{-\alpha} = 1/n^2$. By union bound,

$$\Pr(\text{some elements is not covered}) \leq \sum_{e \in U} \Pr(e \text{ not covered}) \leq n \cdot \frac{1}{n^2} = \frac{1}{n}.$$

This implies w.p. at least $1 - 1/n$, $\mathcal{S}'$ is a covering. ■

In other words, with $\alpha = 2\ln n$, Algorithm 2.4 is correct with probability at least $1 - 1/n$.

Lemma 2.3.4. With $\alpha = 2\ln n$, $\mathcal{S}'$ returned from Algorithm 2.4 has an approximation ratio $4\ln n$ w.p. at least $\frac{1}{2}$.[a]

[a]Note that $\mathcal{S}'$ is not necessary a covering.

Proof. Since $\mathbb{E}[w(\mathcal{S}')] \leq \alpha \sum_i w_i x_i = \alpha\,\mathsf{OPT}_{\mathrm{LP}}$, we have $\Pr(w(\mathcal{S}') \geq 2 \cdot \alpha\,\mathsf{OPT}_{\mathrm{LP}}) \leq 1/2$ from Markov inequality. We see that w.p. $\geq 1/2$, $w(\mathcal{S}') \leq 2 \cdot 2\ln n \cdot \mathsf{OPT}_{\mathrm{LP}} \leq 4\ln n\,\mathsf{OPT}$. ■

Theorem 2.3.2. By running Algorithm 2.4 many times, we get a $(4\ln n)$-approximation algorithm with high probability.[a]

[a]Note that we still need to choose $\mathcal{S}'$.

Proof. Together with Lemma 2.3.3 and Lemma 2.3.4 and using the union bound, the probability of $\mathcal{S}'$ not being a covering or with weight higher than $4\ln n\,\mathsf{OPT}$ is at most $\frac{1}{n} + \frac{1}{2}$, which is less than 1. Hence, by running Algorithm 2.4 many times (independently), the failing possibility is exponential small. ■

Note. With Theorem 2.3.2, we still need to find a valid covering with the lowest cost, where a valid covering with low enough weight is guaranteed to exist with high probability. Note that this is still a polynomial time algorithm since we know that checking $\mathcal{S}'$ is a covering is just linear.

Remark. Indeed, with some smarter algorithm modified from Algorithm 2.4, we can get an H_k approximation ratio.

Lecture 3: Covering-Packing Duality and Primal-Dual Method

2.4 Covering-Packing Duality

7 Sep. 10:30

We first define some useful notions.

Definition 2.4.1 (Strongly independent). Given a set system $(U, \mathcal{S})$, we say $C \subseteq U$ is *strongly independent* if there does not exist $S \in \mathcal{S}$ such that $|C \cap S| \geq 2$.

Remark. Then for any strongly independent set $C \subseteq U$, we know that $\mathsf{OPT}_{\mathrm{SC}} \geq |C|$.[a]

[a] SC denotes set cover.

Now, we're trying to find the **strongest witness** of strongly independent set, which suggests we define the following problem.

Problem 2.4.1 (Maximum strongly independent set). Given a set system $(U, \mathcal{S})$, we want to find the largest strongly independent set.

Remark. For any set system, we have $\mathsf{OPT}_{\mathrm{SIS}} \leq \mathsf{OPT}_{\mathrm{SC}}$.[a]

[a] SIS denotes maximum strongly independent set.

As previously seen (LP dual). Recall how we get the dual of a given LP:

$$\begin{array}{llcll} & \min\ c^{\top} x & \qquad & \max & y^{\top} b \\ & Ax \geq b & & & y^{\top} A \leq c^{\top} \\ (P) & x \geq 0 & & (D) & y \geq 0. \end{array}$$

Also, recall the weak duality ($\mathsf{OPT}_P \geq \mathsf{OPT}_D$) and strong duality ($\mathsf{OPT}_P = \mathsf{OPT}_D$).

Definition 2.4.2 (Covering LP). A primal LP with $A, b, c \geq 0$ is called a *covering LP*.

Definition 2.4.3 (Packing LP). A dual LP with $A, b, c \geq 0$ is called a *packing LP*.

We now give another LP formulation for the unweighted set cover. Given $\mathcal{S} = \{S_1, \ldots, S_m\}$, $U = \{e_1, \ldots, e_n\}$ and define $A \in \mathbb{R}^{n \times m}$ such that

$$A_{ij} = \begin{cases} 1, & \text{if } e_i \in S_j; \\ 0, & \text{otherwise.} \end{cases}$$

Then our LP is defined as

$$\begin{array}{llcll} & \min\ \sum_{j=1}^{m} x_j & \qquad & \max & \sum_{i=1}^{n} y_i \\ & Ax \geq \mathbf{1} & & & y^{\top} A \leq \mathbf{1} \\ (P) & x \geq 0 & & (D) & y \geq 0. \end{array}$$

We see that if we restrict $y_i \in \{0, 1\}$, we see that the dual (D) is just Problem 2.4.1. This can be seen via writing the constraint explicitly:

$$\sum_{i=1}^{n} A_{ij} y_i \leq 1 \Leftrightarrow \sum_{i\colon e_i \in S_j} y_i \leq 1 \text{ for } j \in [m].$$

And indeed, if we look at the weighted version, we have $\sum_{i\colon e_i \in s_j} y_i \leq w(S_j)$.

Now, recall the claim in Theorem 2.2.1, i.e., $y_{e_i} \leq \frac{w(S_j)}{k-i+1}$. We see that the y_{e_i} are just the dual variables in our setup. Additionally, with the observation that we can do this for any set $S = \{e_1, \ldots, e_k\} \in \mathcal{S}$, we have the following lemma.

Lemma 2.4.1. The variable $y' \coloneqq y/H_k$ is dual-feasible, i.e., it's feasible for (D).

Proof. We see that $y_{e_i} \geq 0$ (and hence y_i) trivially, so we only need to show that

$$\sum_{i=1}^{n} A_{ij} y' = \sum_{i=1}^{n} A_{ij} \frac{y_{e_i}}{H_k} \leq w(S_j)$$

for $j \in [m]$. But this is trivial by plugging in $y_{e_i} \leq \frac{w(S_j)}{k-i+1}$ as shown in Theorem 2.2.1, hence

$$\sum_{i=1}^{n} A_{ij} \frac{y_{e_i}}{H_k} \leq \frac{1}{H_k} \sum_{i=1}^{n} A_{ij} \frac{w(S_j)}{k-i+1} \leq \frac{1}{H_k} \sum_{i=1}^{k} \frac{w(S_j)}{k-i+1} = w(S_j),$$

and we're done.[a] ■

[a] Note that in the above derivation, i is kind of overloading, i.e., e_i corresponding to only some i (confusing, but it's how it is...).

With Lemma 2.4.1, we simply run Algorithm 2.1 while maintaining y_e for every e, and we're done.

Theorem 2.4.1. Algorithm 2.1 is an H_k-approximation algorithm in the view of its dual.

Proof. Same as Theorem 2.2.1, but now we have different interpretation. Specifically, if $y' = y/H_k$ is dual-feasible, we know that the corresponding objective value of y' is at most $\mathsf{OPT}_{\mathrm{LP}_D} = \mathsf{OPT}_{\mathrm{LP}_P}$, which is at most $\mathsf{OPT}_{\mathrm{SC}}$ further. Now, since we're dealing with LP, everything is linear includes the objective value, i.e., y is at most $H_k \cdot \mathsf{OPT}_{\mathrm{SC}}$. ■

Remark (Dual fitting). The above method is called *dual fitting*, which is universal as one can easily see. The way to do this is the following.

1. Given an algorithm, distribute the algorithm to $\{y_i\}$.
2. Prove that y/α is dual-feasible.
3. This shows the algorithm is α-approximation algorithm.

2.5 Primal-Dual Method

We first see the general description of the so-called *primal-dual method*.

1. Maintain x (primal solution) and y (dual solution) where x is integral and infeasible, while y is fractional and feasible. Start from $x = y = 0$.
2. **Somehow** increase y until some dual constraints get tight.
3. **Choose** primal variables correspond to tight dual constraints, and update input accordingly.

Remark. We're using dual variables to get a certificate of the lower bound of the optimal problem we're solving.

In terms of set cover, we have the following.

Algorithm 2.5: Set cover – Primal-Dual

Data: A set system $(U, \mathcal{S})$
Result: A covering $\mathcal{S}'$

$\mathcal{S}' \leftarrow \varnothing$, $y \leftarrow 0$
while $U \neq \varnothing$ **do**
  Choose any $e \in U$
  Raise y_e until some constraints get tight
  $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{\text{sets corresponding to tight dual constraints}\}$
  Update U `// Remove newly covered element in` U
return $\mathcal{S}'$

Remark. Algorithm 2.5 is correct and can be implemented efficiently.

Theorem 2.5.1. Algorithm 2.5 is a d-approximation algorithm.

Proof. Firstly, y is feasible. And we see that

$$w(\mathcal{S}') = \sum_{S \in \mathcal{S}'} w(S) = \sum_{S \in \mathcal{S}'} \sum_{e \in S} y_e \leq d \cdot \sum_{e \in U} y_e \leq d \cdot \mathsf{OPT}_{\mathrm{LP}_D} = d \cdot \mathsf{OPT}_{\mathrm{LP}_P} \leq d \cdot \mathsf{OPT}_{\mathrm{SC}}\,.$$

■

Lecture 4: Feedback Vertex Set

2.6 Feedback Vertex Set

12 Sep. 10:30

Following the discussion on primal-dual method, we see another covering problem.

2.6.1 Introduction

We consider the following problem.

Problem 2.6.1 (Feedback vertex set). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a weight function $c\colon \mathcal{V} \to \mathbb{R}^+$, we want to find $F \subseteq \mathcal{V}$ with $\min c(F)$ such that $\mathcal{G}[\mathcal{V} \setminus F]$ has no cycle.[a]

[a] This is equivalent as saying that $\mathcal{G}[\mathcal{V} \setminus F]$ is a forest.

Note (Feedback). The name *feedback* comes from the fact that if there's a cycle in $\mathcal{G}$, then it kind of creates *feedback*.

Note (Edge version). The *edge version* of Problem 2.6.1 can be solved by finding $T \subseteq \mathcal{E}$ be the maximum weight forest,[a] and let $F := \mathcal{E} \setminus T$.

[a] This can be found exactly in polynomial time.

Notation. In this lecture, when talking about cycle, we're referring to the vertices in which. But the meaning can vary from context to context.

Remark. This is a *special case* of Problem 2.1.1.

Proof. Let $\mathcal{C} := \{$set of all (simple) cycles$\}$ and consider Problem 2.1.1 on the set system $(\mathcal{C}, \mathcal{V})$, i.e., we want to find $F \subseteq \mathcal{V}$ such that $\forall C \in \mathcal{C}$, $|F \cap C| \geq 1$. ⊛

Note. The naive algorithm by directly applying methods discussed for Problem 2.1.1, we see that since $\min(\log k, d) = \Omega(n)$ for k being the maximum set size (which is $2^{\Omega(n)}$) and $d = n$, the approximation ratio we can get is $\Omega(n)$, which depends on the size of the input.

Now, the goal in this section is to show the following.

Theorem 2.6.1. There exists a 4-approximation algorithm for Problem 2.6.1.

Remark. Actually, there exists a 2-approximation algorithm.

We also have a hardness of Problem 2.6.1.

Theorem 2.6.2. Achieving $(2 - \epsilon)$-approximation algorithm if NP-hard for all $\epsilon > 0$ assuming the unique games conjecture.

Proof. See Homework 1. ■

2.6.2 Cycle Covering LP

The most natural LP which models Problem 2.6.1 is the so-called *cycle covering LP*, which can be defined as

$$\begin{aligned} \min \ & \sum_{v \in \mathcal{V}} c(v) x_v \\ & \sum_{v \in C} x_v \geq 1 \quad \forall \text{ cycle } C \in \mathcal{C} \\ & x \geq 0, \end{aligned}$$

with the variables being $\{x_v\}_{v \in \mathcal{V}}$ such that $x_v = \mathbb{1}_{v \in F}$.

Remark. We see that this cycle covering LP has $2^{\Omega(n)}$ constraints. But we can actually solve this and get an $O(\log n)$-approximation ratio by smartly rounding the solution.[a] And we can show that this approximation ratio is optimal in terms of this particular LP.

[a] See homework 1.

2.6.3 Density LP

A more sophisticated LP is the so-called *density LP*, defined as

$$\begin{aligned} \min \ & \sum_{v \in \mathcal{V}} c(v) x_v \\ & \sum_{v \in S} x_v (d_v^S - 1) \geq |E(S)| - |S| + 1 \quad \forall S \subseteq \mathcal{V} \\ & x \geq 0 \end{aligned}$$

with the variables being $\{x_v\}_{v \in \mathcal{V}}$.

Notation. The $E(S)$ denotes the edge set in the induced graph $\mathcal{G}[S] = (S, E(S))$, while d_v^S denotes the degree of v in $\mathcal{G}[S]$.

Intuition. The constraint is equivalent as saying that for every induced graph, $\#e \leq \#v - 1$, i.e., we require it to be a forest. Explicitly, $S \subseteq \mathcal{V}$,

$$|E(S)| - \sum_{v \in S} x_v d_v^S \leq |S| - \sum_{v \in S} x_v - 1.$$

Note that in the constraint, the right-hand side is just a lower-bound of $\#e$.

We see that the above LP is not exactly a covering LP since the coefficients can be negative if a set S is not irreducible.

Definition 2.6.1 (Irreducible)**.** The set $S \subseteq \mathcal{V}$ is *irreducible* if for all $v \in S$, v belongs to some cycles in $G[S]$.

Now, it's clear that by looking at $\mathcal{S} = \{S \subseteq \mathcal{V} \mid S \text{ is irreducible}\}$, we have a covering LP defined as

$$\begin{aligned}
\min \ & \sum_{v \in \mathcal{V}} c(v) x_v \\
& \sum_{v \in S} x_v (d_v^S - 1) \geq |E(S)| - |S| + 1 =: b_S \quad \forall S \in \mathcal{S} \\
& x \geq 0.
\end{aligned}$$

We first see why this LP models Problem 2.6.1.

Lemma 2.6.1. The integer version of density LP (denote as IP) is equivalent to Problem 2.6.1.

Proof. If x is feasible for Problem 2.6.1, then x is feasible for the IP. On the other hand, if x is feasible for IP, then for every cycle $C \in \mathcal{C}$, x deletes at least 1 vertex from C. ■

2.6.4 Primal-Dual Method

Now we're ready to solve this LP via primal-dual method. Denote the dual variables as $\{y_S\}_{S \in \mathcal{S}}$, then the dual is

$$\begin{aligned}
\max \ & \sum_{S \in \mathcal{S}} y_S b_S \\
& \sum_{S \ni v} (d_v^S - 1) y_S \leq c(v) \quad \forall v \in \mathcal{V} \\
& y \geq 0.
\end{aligned}$$

Note. For the density LP and its dual, the constraint is still exponentially many, and no one knows how to solve this. But the power of primal-dual method is that we don't really solve this, rather, we just maintain two sets of solutions for both primal and dual. Moreover, we can maintain the primal solution in integral, while the dual solution in fractional.

We now have the following algorithm.

Algorithm 2.6: Feedback vertex set – Primal-Dual

Data: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Result: A minimal feedback vertex set F'

1 $S \leftarrow \mathcal{V}$, $c' = c$, $y \leftarrow 0$ // $c' \in \mathbb{R}^n$ keeps track of slackness of c
2
3 **while** $S \neq \varnothing$ **do**
4 $\quad S \leftarrow \texttt{reduce}(S)$ // Compute $\{v \in S \colon v$ belongs to some cycles in $\mathcal{G}[S]\}$
5 $\quad (\alpha, v) \leftarrow \min_{v\in S} c'(v)/(d_v^S - 1)$[a] // y_S gets tight by increasing unit weight
6 $\quad y_S \leftarrow \alpha$
7 $\quad c'(v) \leftarrow c'(v) - \alpha(d_v^S - 1)$
8 $\quad Z \leftarrow \{v \in S \colon c'(v) = 0\}$
9 $\quad F \leftarrow F \cup Z$, $S \leftarrow S \setminus Z$
10
11 // Compute a minimal feedback vertex set
12 $F' \leftarrow F = \{v_1, \dots, v_\ell\}$ // v_1 is deleted first, v_ℓ is deleted last
13 **for** $i = \ell, \dots, 1$ **do** // reversed greedy
14 $\quad$ **if** $F' \setminus \{v_i\}$ *is a feedback vertex set for* $\mathcal{G}$ **then**
15 $\quad\quad F' \leftarrow F' \setminus \{v_i\}$
16 **return** F'

[a] Note that we also get the argument v.

We see that in Algorithm 2.6, we first use primal-dual method to obtain a feasible feedback vertex set, and then run a reversed greedy algorithm to further ensure we get a good approximation ratio.

Claim. F is a feedback vertex set and y is dual-feasible.

Proof. It should be clear that why F is a feedback vertex set. As for the reason why y is dual-feasible, observe that we have one constraint for each v. After raising y_S for chosen v in line 6 and deduce $c'(v)$ in line 7, v will get removed so the constraint corresponding to v will be satisfied throughout. ⊛

Remark (Reversed greedy). The method we turn F into its minimal is called *reversed greedy*. This just checks that if we remove a vertex v from F' while F' is still feasible, then we just do it. Additionally, we iterate through v in the **reversed** order w.r.t. how v is being added into.

We want to compare the primal cost and the dual cost. The primal cost is

$$c(F) = \sum_{v\in F} c(v) = \sum_{v\in F}\sum_{S\ni v}(d_v^S - 1)y_S = \sum_{S\in\mathcal{S}} y_S \sum_{v\in F\cap S}(d_v^S - 1),$$

while the dual cost is $\sum_{S\in\mathcal{S}} y_S b_S$.

Remark. This is where the primal-dual method is powerful. i.e., by switching the order of summation, if we have some ratio of $\sum_{v\in F\cap S}(d_v^S - 1)$ and b_S for every S, we're done. On caveat is that since S is changing when running Algorithm 2.6, so the final solution F may not be good for this particular S. We need to guarantee some ratio for this F **for all** S.[a]

[a] At least for S with positive y_S.

Lemma 2.6.2. For all $S \in \mathcal{S}$, if F is *minimal* in S,[a] then we have

$$\sum_{v\in S\cap F}(d_v^S - 1) \leq 4\cdot b_S = 4(|E(S)| - |S| + 1).$$

[a] i.e., in $\mathcal{G}[S]$, no $F' \subsetneq F \cap S$ in feedback vertex set.

Proof. Let's first see a simple case.

Intuition. If the graph is 3-regular, then we see that the left-hand side is $\leq 2 \cdot |S|$ by summing over the whole S instead of $S \cap F$, while the right-hand side is $2 \cdot |S| + 4$ since $|E(S)| = 1.5\,|S|$.

This shows that in a 3-regular graph, deleting every vertex in S is actually 4-approximated. And this intuition generalized to general graph with degree greater than 3.

Since we assume S to be irreducible, so we're not interested in degree 0 or 1 vertices (there are no such vertices in an irreducible S). So the only problematic guy is degree-2 vertex. And the only place a degree-2 vertex can live is in a long path.

Figure 2.2: If there are two $v \in F$, by minimality of F, one of v will be strictly unnecessary to break this path in a cycle.

Note. Observe that we only need to delete at most one vertex in any path, and sometimes this may be loose since we can delete one branch node joining two paths, i.e., deleting 1 nodes for two paths.

Let A be the set of degree 2 vertices, and B be the set of vertices with degree larger than 3. Now, consider line segment in the graph. If ℓ is a line segment,

(a) $|F \cap \ell| \leq 1$, i.e., we delete at most one point in ℓ.

(b) If F contains one of the endpoints of ℓ, then $|F \cap \ell| = 0$.

Since F is minimal, the left-hand side is

$$|A \cap F| + \sum_{v \in B \cap F} (d_v^S - 1) \leq \sum_{v \in B \setminus F} d_v^S/2 + \sum_{v \in B \cap F} (d_v^S - 1) \leq \sum_{v \in B} (d_v^S - 1),$$

where the first inequality comes from the fact that if we delete vertices in A, i.e., in the line segment, then we know we don't delete its end points, and by *distributed* that 1 cost into its two end points, each $1/2$.

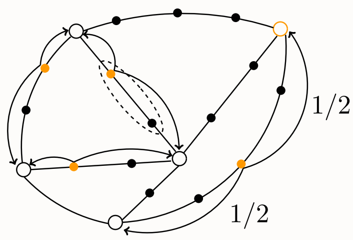


Figure 2.3: Distribute the cost of F.

Similarly, in the right-hand side, the crucial term is

$$|E(S)| - |S| = \sum_{v \in S} (d_v^S/2 - 1) = \sum_{v \in B} (d_v^S/2 - 1)$$

where the last equality holds since for $v \in A$, the summand is just $2/2 - 1 = 0$. It's clear that since $\forall v \in B$, $d_v^S - 1 \leq 4(d_v^S/2 - 1)$, rearranging this inequality gives the result. ■

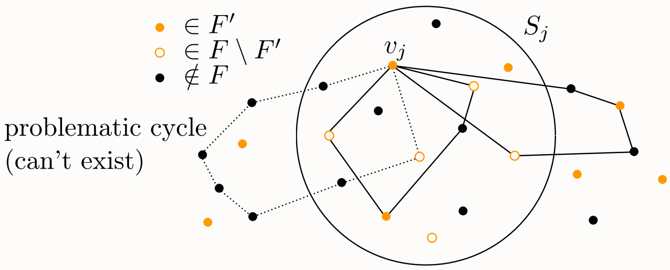
To show Theorem 2.6.1, it's enough to have a minimal F, then the result follows form Lemma 2.6.1. Hence, after obtaining F, Algorithm 2.6 further convert F into F' and try to obtain a minimal version of F. Clearly, F' is still a feedback vertex set, and the minimality of F' is guaranteed by the following lemma.

Lemma 2.6.3. F' is minimal in every S_i, where S_i is the corresponding S in Algorithm 2.6 when v_i is deleted.

Proof. Suppose this is not the case. Then there exists $v_j \in F'$ such that in $\mathcal{G}[S_i]$, $(F \cap S_i) \setminus \{v_j\}$ is still a feedback vertex set in $\mathcal{G}[S_j]$. Notice that we only need to consider the case that $i = j$ since $v_j \in S_i$ means $i \geq j$ from how we order them. In this case, $S_j \subseteq S_i$, hence to check the minimality of F' it's enough to just consider the case that $i = j$. Hence, we consider $(F \cap S_j) \setminus \{v_j\}$ instead.

> **Note.** Here we only consider $G[S_j]$, i.e., we want to say that if v_j is not minimal in $G[S_j]$, then v_j should really be deleted even w.r.t. the whole graph.

Now, observe the following picture in step j of line 13 with cycles contained v_j:



Observe that the middle cycles in $G[S_j]$ must exist from our assumption of $(F \cap S_j) \setminus \{v_j\}$ being still a feedback vertex set, i.e., if a cycle exists in $G[S_j]$, then it must contain another nodes other than v_j that's also in F'. But we see that when we consider cycles outside $G[S_j]$, we have the following.

> **Claim.** No vertices outside S_j which is also in $F \setminus F'$ at step j of line 13
>
> **Proof.** Since S_j is growing, i.e., for $i \leq j$, $S_i \leq S_j$, and we just can't delete something we haven't considered. ⊛

> **Claim.** There are no cycles $C \ni v_j$ such that $C \setminus S_j$ is disjoint from F.
>
> **Proof.** Observe that there are only two ways for a vertex being deleted from the graph, either $v \in Z$, i.e., its dual constraint is tight, or $v \in S$ is deleted since it prevent S being irreducible. Only the latter case will make $v \notin F$, we see that there's no way such a cycle C exists with all vertices outside S_j are preventing s being irreducible, since this cycle C itself is a cycle... ⊛

This implies $F' \setminus \{v_j\}$ is still a feedback vertex set in $\mathcal{G}$ when $i = j$ in Algorithm 2.6 since such a problematic cycle can't exist,[a] which contradicts with the minimality of F'. ■

[a] Explicitly, if this exists, then delete v_j will make F' fail to intersect such a cycle.

Finally, we see that we can prove Theorem 2.6.1.

Proof of Theorem 2.6.1. Firstly, Algorithm 2.6 gives a 4-approximation of the density IP guaranteed by Lemma 2.6.2 and Lemma 2.6.3. Finally, from Lemma 2.6.1, we see that Problem 2.6.1 and the density IP is equivalent, proving the theorem. ■

Chapter 3

Clustering

Lecture 5: Facility Location

3.1 Introduction

14 Sep. 10:30

The problem we're interested in is called the clustering problem.

Problem 3.1.1 (Clustering). Given n objects, partition them into k groups such that

- *Similar* objects are in same group
- *Different* objects are in different group.

Note. We see that Problem 3.1.1 is vague in terms of the definition, which is because this is more like a class of problems. We'll see different notions of *similar* and *different* later when we consider more explicit problems.

In particular, the notion of metric is useful.

Definition 3.1.1 (Metric). Given a set X, a function $d\colon X \times X \to \mathbb{R}^+ \cup \{0\}$ is called a *metric* if

(a) $d(\cdot, \cdot) \geq 0$ and $d(i, j) = 0$ if and only if $i = j$.[a]

(b) $d(i, j) = d(j, i)$ for all $i, j \in X$.

(c) $d(i, j) + d(j, k) \geq d(i, k)$ for all $i, j, k \in X$.

[a]We didn't mention this in lectures, but in math community this should also be included.

Remark (Metric space). Though we didn't formally introduce, but the pair (X, d) of X and a metric d on X is sometimes called a *metric space*.

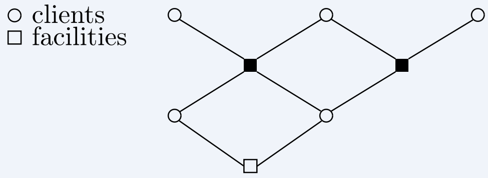
3.2 Facility Location

Let's first look at the problem.

Problem 3.2.1 (Facility location). Given a metric space (X, d) and $P, Q \subseteq X$, $f \in \mathbb{R}^+$ where P is the set of clients, Q is the set of (possible) facilities, we want to open $Q' \subseteq Q$ such that it minimizes $\sum_{i \in P} \min_{j \in Q'} d(i, j) + f\,|Q'|$.[a]

[a]We interpret the first summation as connection cost, the second term as opening cost.

Example. Consider the following example.



If $f = 1$ and we open the black facilities, then the cost is $2 + 5 = 7$ assuming unit length.

We now write down the LP of Problem 3.2.1. Denote variables $\{y_j\}_{j\in Q}$ and $\{x_{ij}\}_{i\in P, j\in Q}$. Then the LP can be written as

$$\begin{aligned}
& \min \sum_{ij} d(i,j)x_{ij} + \sum_j y_j \cdot f \\
& \sum_j x_{ij} \geq 1 && \forall i \in P \quad (\alpha_i) \\
& x_{ij} \leq y_j \Leftrightarrow y_j - x_{ij} \geq 0 && \forall i, j \quad (\beta_{ij}) \\
(P) \quad & x, y \geq 0.
\end{aligned}$$

Denote the dual variables as α_i and β_{ij}, the dual is

$$\begin{aligned}
& \max \sum_i \alpha_i \\
& \alpha_i - \beta_{ij} \leq d(i,j) && \forall i, j \quad (x_{ij}) \\
& \sum_i \beta_{ij} \leq f && \forall j \quad (y_i) \\
(D) \quad & \alpha, \beta \geq 0.
\end{aligned}$$

Remark. If (α, β) is feasible, redefine $\beta_{ij} := \max(0, \alpha_i - d(i,j))$, it's still feasible and will not affect the objective value. We see that we can drop β and only look at α.

We can then define the following useful notion called cluster.

Definition 3.2.1 (Cluster). A *cluster* $C := (j, P')$ is the order pair for $j \in Q$ and $P' \subseteq P$, where the cost $c(C)$ is calculated by directing all $i \in P'$ to j, i.e., $c(C) = f + \sum_{i\in P'} d(i,j)$.

Notation. We denote the set of all clusters C by $\mathcal{C}$.

Remark (Just set cover!). We see that Problem 3.2.1 is equivalent to set cover on $(P, \mathcal{C})$.

Proof. If we write down the LP for set cover on $(P, \mathcal{C})$, we have

$$\begin{aligned}
& \min \sum_{C\in\mathcal{C}} c(C) \cdot y_C && \qquad \max \sum_{i\in P} \alpha_i \\
& \sum_{C\ni i} y_C \geq 1 \quad \forall i \in P && \qquad \sum_{i\in C} \alpha_i \leq c(C) \quad \forall C \in \mathcal{C} \\
(P) \quad & y \geq 0 && \qquad (D) \quad \alpha \geq 0,
\end{aligned}$$

which is equivalent to what we have as above. ⊛

But observe that the number of clusters is $|Q| \cdot 2^{|P|}$, hence directly solve either (P) or (D) is not feasible. In this case, we can use the primal-dual method.

3.2.1 Primal-Dual Method

Let's first see the primal-dual algorithm on (P) and (D) derived above.

Algorithm 3.1: Facility location – Primal-Dual

Data: A set of clients $P \subseteq X$, a set of (possible) facilities $Q \subseteq X$, facility cost f
Result: A set of opened facilities $Q' \subseteq Q$

1 $S \leftarrow \varnothing$, $Q' \leftarrow \varnothing$, $\alpha \leftarrow 0$ // S:connected clients, O:open facilities
2
3 **while** $S \neq P$ **do**
4   **while** *True* **do**
5   increase all $\{\alpha_i\}_{i \in P \setminus S}$ by a *unit*
6   **if** *some* $j \in Q \setminus Q'$ *s.t.* $\sum_{i \in P} \beta_{ij} = f$ **then** // j gets tight (open)
7   **break**
8   **else if** *some* $i \in P \setminus S$ *s.t.* $\alpha_i \geq d(i, j)$ **then** // i can connect to $j \in Q'$
9   **break**
10     $Q' \leftarrow \{\text{tight facilities}\}$ // Update Q'
11     $S \leftarrow \{\text{clients connected to } Q'\}$ // Update S
12
13 // Trim down Q'
14 $G = (Q', E \coloneqq \{(j, j') \colon \exists i \in P \text{ such that } \alpha_i > d(i, j), \alpha_i > d(i, j'), j, j' \in Q'\})$
15 Compute Q'' s.t. $\forall j \in Q'$, either $j \in Q''$ or $\exists j' \in Q''$ s.t. $(j, j') \in E$ // max independent set
16 **return** Q''

Note. line 6 and line 8 can happen in the same time.

Intuition. We're basically increasing the cost i willing to pay and stop (in the second while loop) when i finally connect to j. Or one can also interpret α_i as the time i connects to some facilities j.

This directly relates to the fact that for all i, j, if i, j are connected, then $d(i, j) \leq \alpha_i$, which is exactly the spirit of the primal-dual method since we want to argue the upper-bound in terms of α. But before that, we need to argue that α is actually feasible in order to make this bound valid.

Lemma 3.2.1. α is dual-feasible in Algorithm 3.1.

Proof. Firstly, α start from 0 which is feasible. Now, for α_i violates the constraints $\sum_{i \in C} \alpha \leq c(C) = f + \sum_{i \in P'} d(i, j)$, there are two possibilities, but both are handled in Algorithm 3.1. Specifically, line 6 and line 8:

- In line 6: This corresponds to some j gets opened, we then need to make sure that no α_i will pay toward j for its open cost f. But this is clear since whoever i is paying non-zero amounts to j for its f, i immediately connect to j and will be clicked out from $P \setminus S$, meaning that their dual α_i will not be increased anymore.
- In line 8: This corresponds to when i want to connect (willing to pay non-zero amount to) an already opened j. But we see that whenever i willing to pay for an already opened j, we immediately connect them and so j gets nothing (hence will not be violated) while i just pays for the distance to go to j.

In all, throughout Algorithm 3.1, α is feasible. ■

Note (Trim down). Just like Algorithm 2.6, after getting the initial solution Q', we'll soon see in the analysis section that it's kind of wasteful, so we trim it down to obtain a better solution.

3.2.2 Analysis

We first do a naive analysis, i.e., try to bound the connected cost and opening cost for Q' obtained in Algorithm 3.1 before line 12, which turns out to be not working. The problem is not on connected cost, since as noted above, $d(i, j) \leq \alpha_i$ so the connection cost is at most $\sum_i \alpha_i$.

Remark. Bound the opening cost naively can't guarantee a constant approximation factor.

Proof. To bound opening cost, we see that

$$\text{opening cost } = f\left|Q'\right| = \sum_{j\in Q'} f = \sum_j \sum_i \beta_{ij} = \sum_i \sum_{j\in Q'} \beta_{ij}.$$

Observe that since $\beta_{ij} = \max(0, \alpha_i - d(i,j)) \leq \alpha_i$, hence if we can guarantee for each i, it only pays for one j, then we will get a 2-approximation. But this might not be the case since we don't have control of how many j that i is paying. ⊛

Let's first introduce some notions in order to analyze Algorithm 3.1.

Notation (Connecting witness)**.** The first open facility connected to i is called the *connecting witness* $w(i) \in Q$ for every $i \in P$.

Notation (Contributing)**.** We say (i,j) is *contributing* if $\alpha_i > d(i,j)$, i.e., $\beta_{ij} > 0$.[a]

[a] We now have a strict inequality, i.e., i is now paying some non-trivial amount to j.

Note that the problem in the naive solution happens when a client i pays multiple facilities j. And a simple idea is to close some facilities j such that every client pay at most 1 facility.

Intuition. If i is contributes to two facilities j and j', we close down one of them basically since this is where the problem comes from. This is exactly how we trim down Q': by considering $G = (Q', E)$ such that $(j,j') \in E$ if and only if $\exists i \in P$ that contributes to both j and j', taking maximal independent set of G exactly makes i paying to only one j.

Note. In this case, we take care of opening cost, but the connected cost might be worse, so we basically turn to bound another quantity while still keep one term simple to bound.

Notation (Directed connected)**.** We say $i \in P$ is *directed connected* if $j \in Q''$ such that (i,j) is connected ($\alpha_i \geq d(i,j)$). For these i, divide α_i into $\alpha_i^f := \beta_{ij}$ and $\alpha_i^c := d(i,j)$, i.e., $\alpha_i = \alpha_i^f + \alpha_i^c$.

Notation (Indirected connected)**.** We say i is *indirectly connected* if i is not directed connected,[a] and like in directed connected, $\alpha_i =: \alpha_i^f + \alpha_i^c$ where $\alpha_i^f = 0$, $\alpha_i^c = \alpha_i$.

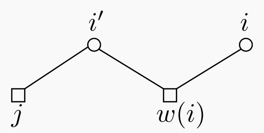


Figure 3.1: When i is indirected connected.

[a] i.e., there exists j such that $(j, w(i)) \in E$, hence there exists i' such that (i', j) and $(i', w(i))$ contributing.

Now, we can bound the opening cost $f\left|Q''\right|$ for Q'' more carefully. It's now

$$f\left|Q''\right| = \sum_{j\in Q''}\sum_i \beta_{ij} = \sum_{j\in Q''}\sum_{\text{d.c. } i} \beta_{ij} = \sum_{\text{d.c. } i}\left[\sum_{j\in Q''} \beta_{ij}\right] = \sum_{\text{d.c. } i} \alpha_i^f.$$

As for connected cost, we see that if i is directed connected, $d(i,j) \leq \alpha_i^c$, while if i is indirected connected, it's not so clear. However, we have the following.

Claim. If i is indirected connected, then $d(i,j) \leq 3\alpha_i$.

Proof. Note that $(j, w(i)) \in E$ and $d(i,j) \leq \alpha_i + 2\alpha_{i'}$ by looking at Figure 3.1, hence it's sufficient to prove $\alpha_{i'} \leq \alpha_i$. To do this, for some facility ℓ, define t_ℓ to be the time ℓ open in line 6, and α_i be the time i connected in line 8. We see that

- If (i, ℓ) are contributing, then $\alpha_i \leq t_\ell$.
- If $\ell = w(i)$, then $t_\ell \leq \alpha_i$.

Combining these together, we have $\alpha_{i'} \leq t_{w(i)} \leq \alpha_i$. ⊛

Finally, we have the following.

Theorem 3.2.1. Algorithm 3.1 is a 3-approximation algorithm.

Proof. The cost of Q'' produce by Algorithm 3.1 is just the connected cost of plus the opening cost of Q'', which can be bounded as

$$\text{final cost} = \text{connected cost} + \text{opening cost} \leq \sum_i 3\alpha_i^c + \sum_i \alpha_i^f \leq 3\sum_i \alpha_i \leq 3\,\mathsf{OPT},$$

which shows that it is a 3-approximation algorithm. ■

Note. Notice that in the above proof, since we know that the opening cost is exactly $\sum_i \alpha_i^f$, and hence even if we pay 3 times of the opening cost, we still get a 3-approximation algorithm.

Remark. Algorithm 3.1 is a very basic algorithm which can be used even as a black-box for other clustering problems. We'll revisit this later and consider other metrics and see what can we improve.

Lecture 6: Facility Location with LMP Approximation

3.2.3 Hardness

19 Sep. 10:30

For Problem 3.2.1, we have the following.

(a) 1.488-approximation [Li13]

(b) 1.463-approximation is NP-hard [GK99]

Turns out that specifically for Problem 3.2.1, we can have a more refine notion of approximation ratio defined below.

Definition 3.2.2 (LMP approximation). An algorithm ALG which solves facility location is called *γ-Lagrangian multiplier preserving approximation* (LMP-approximation) if

$$\frac{\text{conn(ALG)}}{\gamma} + \text{open(ALG)} \leq \sum_i \alpha_i$$

for some $\gamma > 0$.[a]

[a]The opening cost is just $k'f$ if ALG opens k' centers.

Remark. The notion of LMP approximation is due to Lagrangian multiplier in the field of optimization, where the dual variables are treated as a Lagrangian multipliers. And Definition 3.2.2 says that we're not approximating $k'f$ at all, hence it's *preserving*.

And indeed, we now have a more refined characterization about Algorithm 3.1.

Corollary 3.2.1. Algorithm 3.1 is a 3-LMP approximation algorithm.

Remark (SOTA). If we look at the SOTA result in terms of LMP, we have the following.

(a) 3-LMP approximation [JV01]

(b) 2-LMP approximation [JMS02]

(c) 1.99 . . . 9-LMP approximation [Coh+22]

(d) 1.73-LMP approximation[a] is NP-hard [JMS02]

[a] The number comes from $1 + 2/e$.

3.2.4 Greedy Method

Let's take another look at Problem 3.2.1 and see it as an instance of Problem 2.1.1 where the universe is all the clients P, while the collection of sets are pairs of facility and its connected clients, i.e., clusters. Then, it's natural to consider using a similar algorithm as Algorithm 2.1 to solve this.

Algorithm 3.2: Facility location – Greedy

Data: A set of clients $P \subseteq X$, a set of (possible) facilities $Q \subseteq X$, facility cost f[a]
Result: A set of opened facilities $Q' \subseteq Q$

$S \leftarrow \varnothing$, $Q' \leftarrow \varnothing$
while $S \neq P$ **do**
  choose $(j, T) \in Q \times \mathcal{P}(P \setminus X)$ with minimum $c((j, T)) / |T|$
  $Q' \leftarrow Q' \cup \{j\}$
  $S \leftarrow S \cup T$
return Q'

[a] We didn't use it explicitly in the algorithm since we hide it in the cost function $c(\cdot)$.

This is just Algorithm 2.1, hence we have H_n-approximation. But as we have seen in Theorem 3.2.1, we have achieved a constant approximation ratio for Problem 3.2.1. Hence, we should be able to do better based on Algorithm 3.2.

Remark. If we modify Algorithm 3.2 such that for all (j, T), if j is open, then we define the cost of this cluster as

$$c((j, T)) := \frac{\sum_{i \in T} d(i, j)}{|T|}.$$

We'll achieve 1.861-approximation, but the analysis is complex.

Instead, we're going to see other variations based on Algorithm 3.2.

First Modification

We see observe that $c((j, T)) / |T|$ is increasing in Algorithm 3.2. Also, if $\alpha := c((j, T)) / |T|$, then for all $i \in T$, $d(i, j) \leq \alpha$ where we interpret this as i *pays* α_i to cover the connection cost $d(i, j)$ and the opening cost $\alpha_i - d(i, j)$ of j. Following this intuition, if we change line 6 in Algorithm 3.1 (with only first phase) such that the summation is over $P \setminus S$, it becomes exactly Algorithm 3.2. proof!

Algorithm 3.3: Facility location – Greedy Modification I

Data: A set of clients $P \subseteq X$, a set of (possible) facilities $Q \subseteq X$, facility cost f
Result: A set of opened facilities $Q' \subseteq Q$

1 $S \leftarrow \varnothing$, $Q' \leftarrow \varnothing$, $\alpha \leftarrow 0$ // S:connected clients, Q':open facilities
2
3 **while** $S \neq P$ **do**
4   **while** *True* **do**
5   increase all $\{\alpha_i\}_{i \in P \setminus S}$ by a *unit*
6   **if** *some* $j \in Q \setminus Q'$ *s.t.* $\sum_{i \in P \setminus S} \beta_{ij} = f$ **then** // j gets tight (open)
7   **break**
8   **else if** *some* $i \in P \setminus S$ *s.t.* $\alpha_i \geq d(i,j)$ **then** // i can connect to $j \in Q'$
9   **break**
10   $Q' \leftarrow Q' \cup \{j\}$ // Update Q'
11   $S \leftarrow S \cup \{i \in P \setminus S \colon \alpha_i \geq d(i,j)\}$ // Update S
12 **return** Q'

> **Remark.** Since line 6 and line 8 can happen simultaneously, while what we just said assumes the opposite, so we need to further modify Algorithm 3.1 in line 10 and line 11.

Second Modification

Another potential modification gives us a 1.61-approximation. We essentially allow $i \in S$ to *switch* in Algorithm 3.3, i.e., after i connects to j, if j' is closer to i later, i can *offer* with $d(i,j) - d(i,j')$ to other facilities.

Algorithm 3.4: Facility location – Greedy Modification II

Data: A set of clients $P \subseteq X$, a set of (possible) facilities $Q \subseteq X$, facility cost f
Result: A set of opened facilities $Q' \subseteq Q$

1 $S \leftarrow \varnothing$, $Q' \leftarrow \varnothing$, $\alpha \leftarrow 0$ // S:connected clients, Q':open facilities
2
3 **while** $S \neq P$ **do**
4   **while** *True* **do**
5   increase all $\{\alpha_i\}_{i \in P \setminus S}$ by a *unit*
6   **if** *some* $j \in Q \setminus Q'$ *s.t.* $\sum_{i \in S} (d(i, w(i)) - d(i,j))^+ + \sum_{i \in P \setminus S} \beta_{ij} = f$[a] **then**
7   **break**
8   **if** *some* $i \in P \setminus S$ *s.t.* $\alpha_i \geq d(i,j)$ **then** // i can connect to $j \in Q'$
9   **break**
10   $Q' \leftarrow Q' \cup \{j\}$ // Update Q'
11   $S \leftarrow S \cup \{i \in P \setminus S \colon \alpha_i \geq d(i,j)\}$ // Update S
12 **return** Q'

[a] We define $a^+ := \max(a, 0)$ and also, $w(i)$ is now the *current* facility i is connected to.

Third Modification

If we run Algorithm 3.4 with facility cost being $\hat{f} := 2f$, we can have a 2-LMP approximation algorithm as follows.

Algorithm 3.5: Facility location – Greedy Modification III

Data: A set of clients $P \subseteq X$, a set of (possible) facilities $Q \subseteq X$, facility cost f
Result: A set of opened facilities $Q' \subseteq Q$

$S \leftarrow \varnothing$, $Q' \leftarrow \varnothing$, $\alpha \leftarrow 0$ // S:connected clients, Q':open facilities

while $S \neq P$ **do**
  while *True* **do**
    increase all $\{\alpha_i\}_{i \in P \setminus S}$ by a *unit*
    if *some* $j \in Q \setminus Q'$ *s.t.* $\sum_{i \in S}(d(i, w(i)) - d(i,j))^+ + \sum_{i \in P \setminus S} \beta_{ij} = \hat{f}$[a] **then**
      break
    if *some* $i \in P \setminus S$ *s.t.* $\alpha_i \geq d(i,j)$ **then** // i can connect to $j \in Q'$
      break
  $Q' \leftarrow Q' \cup \{j\}$ // Update Q'
  $S \leftarrow S \cup \{i \in P \setminus S \colon \alpha_i \geq d(i,j)\}$ // Update S
return Q'

[a] We define $a^+ := \max(a, 0)$ and also, $w(i)$ is now the *current* facility i is connected to.

It's clear that in Algorithm 3.5, the connection cost plus 2 times the opening cost is $\sum_{i \in P} \alpha_i$ from how we design the algorithm by changing the facility cost from f to $\hat{f} := 2f$. Now, a crucial lemma is the following.

Lemma 3.2.2. (α', β') is dual feasible, where $\alpha' := \alpha/2$, $\beta'_{ij} := (\alpha'_i - d(i,j))^+$.

Proof. It's sufficient to consider $j \in Q$ and prove that $\sum_{i \in P'} \beta'_{ij} \leq f$ where $P' := \left\{ i \colon \beta'_{ij} > 0 \right\} = [n]$ where we're overloading n here. Let's order α_i such that $\alpha_1 \leq \cdots \leq \alpha_n$ where α_i is the time i when i is *first connected.*

Claim. For all $i, k \in P'$ such that $\alpha_k \leq \alpha_i$, at time (right before) α_i, offer from k to j[a] is at most $\alpha_i - d(i,j) - 2d(k,j)$ for any $j \in Q$.

[a] We assume k currently (or is going to) connects to j'.

Proof. We see that if $\alpha_i = \alpha_k$, the offer is just $(\alpha_k - d(i,j))^+$. Otherwise, we have $\alpha_k < \alpha_i$. If $\alpha_i > d(k,j') + d(k,j) + d(i,j)$, we immediately get a contradiction since from triangle inequality, $\alpha_i > d(i,j')$, i.e., i already connect to j'. Hence,

$$\alpha_i \leq d(k,j') + d(k,j) + d(i,j).$$

Then, the offer from k to j is $(d(k,j') - d(k,j))^+ \geq \alpha_i - d(i,j) - 2d(k,j)$. ⊛

Observe that for all $i \in [n]$, we have

$$\sum_{k=1}^{i-1}(\alpha_i - d(i,j) - 2d(k,j)) + \sum_{k=i}^{n}(\alpha_i - d(k,j)) \leq \hat{f} \tag{3.1}$$

by considering the total offer from k to j at time (right before) α_i. Now, we add Equation 3.1 for all $i \in [n]$, we have

$$n\sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}(i-1)d(i,j) - 2\sum_{k=1}^{n}(n-k)d(k,j) - \sum_{i=1}^{n} k \cdot d(k,j) \leq n\hat{f} = 2nf.$$

Since the summation over k is just indexes, we can change it to i, hence

$$\begin{aligned}2nf &\geq n\sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}(i-1)d(i,j) - 2\sum_{i=1}^{n}(n-i)d(i,j) - \sum_{i=1}^{n}i\cdot d(i,j)\\ &\geq n\sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}id(i,j) - 2\sum_{i=1}^{n}(n-i)d(i,j) - \sum_{i=1}^{n}i\cdot d(i,j) = n\sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}2nd(i,j)\end{aligned}$$

where we turn the factor $(i-1)$ into i and gather the terms together. Clean up a bit, we have

$$n\sum_{i=1}^{n}\alpha_i - 2n\sum_{i=1}^{n}d(i,j) \leq 2nf \Leftrightarrow \frac{\sum_{i=1}^{n}\alpha_i}{2} - \sum_{i=1}^{n}d(i,j) \leq f,$$

finishing the proof. ■

From Lemma 3.2.2, we immediately have the following.

Theorem 3.2.2. Algorithm 3.5 is a 2-LMP approximation algorithm w.r.t. the original f.

Lecture 7: k-Median and LMP Approximation

3.3 k-Median

21 Sep. 10:30

Let's look at another clustering problem.

Problem 3.3.1 (k-median)**.** Given a metric space (X, d) and $P, Q \subseteq X$ with $k \in \mathbb{N}$, find $Q' \subseteq Q$ with $|Q'| = k$ which minimizes $\sum_{i\in P}\min_{j\in Q'} d(i,j)$.

The natural linear programming for Problem 3.3.1 is the following. Consider $\{x_{ij}\}_{i\in P, j\in Q}$ and $\{y_j\}_{j\in Q}$, then

$$\begin{aligned}\min\ & \sum_{ij} x_{ij}d(i,j) & & \\ & \sum_{j} x_{ij} \geq 1 & \forall i\in P \quad & (\alpha_i)\\ & x_{ij}\leq y_j & \forall i\in P, j\in Q \quad & (\beta_{ij})\\ & \sum_j y_j \leq k & & (f)^{1}\\ & x, y\geq 0 & & \end{aligned}$$

Intuition. We interpret x_{ij} as follows: if $x_{ij} = 1$, then i belongs to j. And $y_j = 1$ if j is the actual median we choose (i.e., in Q'). As for constraints, both $\sum_j x_{ij}\geq 1$ and $\sum_j y_j \leq k$ are clear, while for $x_{ij}\leq y_j$, we see that it can't be the case that $x_{ij} = 1$ while $y_j = 0$, i.e., we can't have the case that x_{ij} belongs to j while j isn't even in Q'.

The dual is then

$$\begin{aligned}\max\ & \sum_i \alpha_i - kf & \\ & \sum_i \beta_{ij} \leq d(i,j) & \forall i\in P, j\in Q\\ & \sum_i \beta_{ij}\leq f & \forall j\in Q\\ & \alpha,\beta\geq 0 & \end{aligned}$$

[1]Notice that compare to Problem 3.2.1, f here is a variable but not a given facility cost! The reason why we do this will be clear soon.

Note. Notice that this is exactly the dual as Problem 3.2.1, except that we now have an additional $-kf$ term in the objective function. Although f is not included in the statement of Problem 3.3.1, by denoting one of the dual variable f, we get a similar formulation compare to Problem 3.2.1.

Due to the similarity between Problem 3.3.1 and Problem 3.2.1, we can try to use Algorithm 3.5 which solves Problem 3.2.1 with 2-LMP guarantee. But note that in Problem 3.2.1, we need to specify f. Suppose we guessed f, and we run a γ-LMP approximation algorithm and somehow get $k' = k$. Then we have

$$\frac{\mathrm{conn}(\mathrm{ALG})}{\gamma} \leq \sum_i \alpha_i - kf \leq \mathsf{OPT}_{k\text{-med.}},$$

i.e., this is a γ-approximation algorithm. So now, the task is to guess f such that the algorithm gives exactly k centers.

3.3.1 Bipoint Solution

Turns out that we don't have ideas about the relation between k and f, the only thing we know is that if $f \to \infty$, k decreases, other than that it behaves quite arbitrary.

Remark. The relation between k and f indeed highly depends on what algorithm we use. But at least for Algorithm 3.5, nobody knows anything in this case.

Given this fact, just randomly guess one f doesn't work. A new idea is then to maintain two solutions (or interval) $[f^2, f^1]$ such that $f^2 \leq f^1$,[2] where

- at f^2, the algorithm opens $k^2 \geq k$ facilities;
- at f^1, the algorithm opens $k^1 \leq k$ facilities.

Then, a naive approach is to use binary search and get $f^2 \leq f^1$ such that

$$\left|f^1 - f^2\right| \leq \frac{\epsilon\,\mathsf{OPT}}{n}.$$

Notice that the whole point of doing binary search is because we assume that if $k^2 \geq k$ at f^2 and $k_1 \leq k$ at f^1, then we can find an $f^* \in [f^2, f^1]$ such that we get exactly $k^* = k$ at f^*.

Remark (Caveat of achieving k)**.** This is probably not the case for Algorithm 3.2 (2-LMP) since the decision is quite sequential; but if we use Algorithm 3.1 (2-LMP), since there are lots of maximal independent sets, so by doing **a lot more work**, we can actually achieve this.

Now, assume that we have continuity of the relation between k and f by carefully designing our (γ-LMP) algorithm, then $\exists a \in [0,1]$ and $b \coloneqq 1-a$ such that $k \coloneqq ak^1 + bk^2$ where $k^1 \leq k \leq k^2$. Denote C^i as the connection cost $\mathrm{conn}(f^i)$ of f^i such that $C^1 \geq C^2$, we have

$$\begin{cases} C^1 + \gamma k^1 f^1 \leq \gamma \sum_i \alpha_i^1, & (\times a) \\ C^2 + \gamma kr2 f^2 \leq \gamma \sum_i \alpha_i^2, & (\times b) \end{cases}$$

hence,

$$aC^1 + bC^2 \leq \gamma\left(a\sum_i \alpha_i^1 + b\sum_i \alpha_i^2 - ak^1f^1 - bk^2f^2\right) \leq \gamma \underbrace{\left(\sum_i \alpha_i - kf\right)}_{\leq \mathsf{OPT}_{k\text{-med.}}} + \underbrace{\gamma k\left|f^1 - f^2\right|}_{\leq \epsilon\,\mathsf{OPT}_{k\text{-med.}}},$$

where we set $\alpha \coloneqq a\alpha^1 + b\alpha^2$ and $f \coloneqq \max(f^1, f^2)$.

[2] We start from $f^2 = 0$ and $f^1 = \infty$, where we set f^1 arbitrary large.

Note. To make sure $\sum_i \alpha_i - kf \leq \mathsf{OPT}_{k\text{-med.}}$, we need to check that (α, f) is dual-feasible for Problem 3.3.1.

Proof. The feasibility comes from the fact that the first two constraints of Problem 3.3.1 are linear, so they're automatically satisfied. The only non-trivial constraint is $\sum_i \beta_{ij} \leq f$, but since we choose f to be the maximum, it'll be more satisfied. ⊛

Definition 3.3.1 (Bipoint solution). Given F^1, F^2 with $\lvert F^1 \rvert = k^1$, $\lvert F^2 \rvert = k^2$ and $k = ak^1 + bk^2$ for $a, b \in [0, 1]$ and $a + b = 1$, the *bipoint solution*, denoted as $aF^1 + bF^2$, satisfies

$$aC^1 + bC^2 \leq \gamma \cdot \mathsf{OPT}_{k\text{-med.}}.$$

3.3.2 Bipoint Rounding

From Definition 3.3.1, it's natural to do the so-called bipoint rounding.

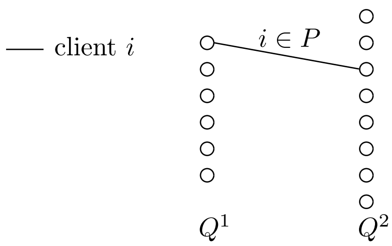
Definition 3.3.2 (δ-bipoint rounding). Given solutions F^1 and F^2, a solution F with $\lvert F \rvert = k$ such that

$$\operatorname{conn}(F) \leq \delta \cdot (aC^1 + bC^2) = \delta \cdot \operatorname{conn}(aF^1 + bF^2)$$

is the so-called *δ-bipoint rounding solution.*

Note. If we have a δ-bipoint rounding of a γ-LMP algorithm solution, then we automatically have an approximation ratio of $\delta \cdot \gamma$ for this bipoint rounding solution.

Back to Problem 3.3.1, we see that we can actually get a 2-bipoint rounding as follows. Consider we create a bipartite graph with $Q^1, Q^2 \subseteq Q$ being two sides of the graph. Then for each $i \in P$, i is connected to the closest facility in Q^1, and also another closest facility in Q^2, so we can create an edge between these two facilities.



Now, for a fixed $i \in P$, let $d_j := d(i, Q^j)$ for $j = 1, 2$, we want to compare our designed final cost to $aC^1 + bC^2$, so for this fixed i, we want to make sure i *pays* not much more than $ad_1 + bd_2$.

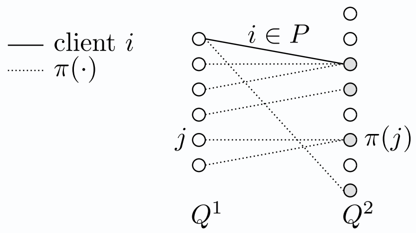
Intuition. We see that a natural rounding algorithm is the following: for an $i \in P$, if its closest facility in Q^1 is opened while its closest facility in Q^2 is not opened, we may just direct i to the opened one in Q^1, same for the other case. Now, if both facilities are opened, then we direct i to the facility in F^1 with probability a, while to the facility in Q^2 with probability $1 - a = b$.

Remark. The problem of the above algorithm is that we don't have control about the total number of the final open facilities: it can be the case that at the end we open every facility in Q^2, which is k^2, not k. So we need to sometimes direct i to other facilities (in Q^1) that is not closest to which.

For $j \in Q^1$, let $\pi(j)$ be the closest facility in Q^2 to j, and let Q^* be the image of such a map π, i.e., $Q^* = \left\{ j' \in Q^2 \colon j' = \pi(j) \text{ for some } j \in Q^1 \right\}$.

Note. We may assume $\lvert Q^* \rvert = k^1$.

Proof. Clearly, $|Q^*| \leq k^1$. And if $|Q^*| < k^1$, we add arbitrary centers so that $|Q^*| = k^1$.



For example, the initial image size above is only 4, we need to add 2 more arbitrary centers into Q^*. ⊛

To open the facilities as what we want, consider the following rounding algorithm.

Algorithm 3.6: k-Median – 2-Bipoint Rounding

Data: A set of clients $P \subseteq X$, a set of (possible) facilities $Q \subseteq X$, $a \in (0,1)$, $\epsilon \in (0,1)$, $k \in \mathbb{N}$
Result: A set of opened facilities $Q' \subseteq Q$ with $|Q'| = k$

1 $(Q^1, Q^2) \leftarrow$`binary-search`$(P,\, Q,\, \epsilon)$ `// achieve` $\left|f^1 - f^2\right| \leq \epsilon\,\mathsf{OPT}\,/n$
2
3 $Q' \leftarrow \varnothing$, $k^1 \leftarrow \left|Q^1\right|$, $k^2 \leftarrow \left|Q^2\right|$, $Q^* \leftarrow \left\{j' \in Q^2 \colon j' = \pi(j) \text{ for some } j \in Q^1\right\}$
4
5 **for** $j \in Q^1$ **do**
6   **if** `rand`$(0,\, 1) \leq a$ **then** `// open` Q^1 `w.p.` a
7         $Q' \leftarrow Q' \cup \{j\}$
8   **else** `// open` Q^* `w.p.` $1-a$
9         $Q' \leftarrow Q' \cup \{\pi(j)\}$
10
11 `// still need to open` $k - k^1$ `more`
12 $Q' \leftarrow Q' \cup \left\{(k-k^1) \text{ random } j \in Q^2 \setminus Q^*\ \right\}$
13 **return** Q'

Remark. Algorithm 3.6 is a randomized algorithm which will **always** open k facilities. The randomness comes from the cost, i.e., we can analyze its cost in expectation.

Intuition. Algorithm 3.6 is kind of *mimicking* what we want, since

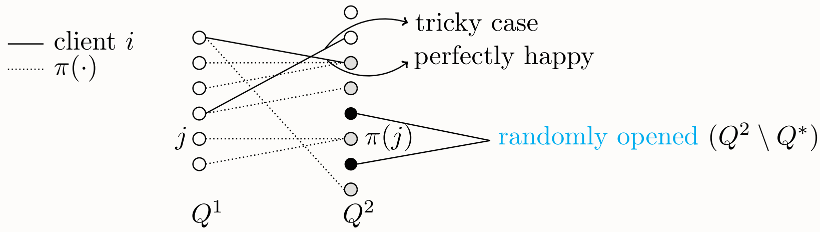
- $j \in Q^1$, $\Pr(j \text{ open}) = a$
- $j \in Q^*$, $\Pr(j \text{ open}) = 1 - a = b$
- $j \in Q^2 \setminus Q^*$, $\Pr(j \text{ open}) = \frac{k-k^1}{k^2-k^1} = b$

Theorem 3.3.1. Algorithm 3.6 is a 2-bipoint algorithm (in expectation).

Proof. Let's analyze a bit careful. Fixing an $i \in P$, and denote its closest facility in Q^1 as j^1, and the closest facility in Q^2 as j^2. If j^1 is not opened, then we know $\pi(j^1)$ is opened for sure in line 8. We see that

- If j^2 is in Q^*, then we know i will be direct to either j^1 or j^2 in line 5, i.e., i is perfectly happy since it can go to one of the closest facility.
- The tricky case is when j^2 is not in Q^*.
  - If j^1 is opened, i can still go to j^1 without problem.
  - If j^1 is also not opened, we know that $\pi(j^1)$ will be opened in line 8. In this worst case,

we just direct i to $\pi(j^1)$ and the distance will be $i \to j^1 \to \pi(j^1)$, which is bounded by $d_1 + d(j^1, \pi(j^1))$. But observe that $d(j^1, \pi(j^1)) \leq d_1 + d_2$, so we have $2d_1 + d_2$.



In all, we have the following.[a]

	Distance	Probability
j^2 open	d_2	b
j^2 not open, j^1 open	d_1	$\geq (a-b)^+ =: M$
none of j^1, j^2 open	$2d_1 + d_2$	$\leq 1 - b - M$

Then, the expected cost is just[b]

$$\mathbb{E}\left[i\text{'s connection cost}\right] \leq bd_2 + Md_1 + (1-b-M)(2d_1 + d_2),$$

and we now have two cases.

- If $b \geq a$, then $b \geq 1/2$, $M = 0$ and

$$\mathbb{E}\left[i\text{'s connected cost}\right] \leq b \cdot d_2 + (1-b)(2d_1 + d_2) = 2ad_1 + d_2 \leq 2(ad_1 + bd_2).$$

- If $a > b$, then $a > 1/2$, $M = a - b$ and

$$\mathbb{E}\left[i\text{'s connected cost}\right] \leq b \cdot d_2 + (a-b)d_1 + b(2d_1 + d_2) = d_1(a+b) + d_2(b+b) \leq 2(ad_1 + bd_2).$$

This shows Algorithm 3.6 is a 2-bipoint algorithm in expectation, proving the result. ■

[a] This is a slightly worse result since we force i to go to j^2 if j^2 is opened, but actually, i can go to j^1 if j^1 is opened too with shorter distance. But this still gives us a good enough bound.

[b] Since the final case is always worse than the second case, it is legal to assume that the second case has the minimum probability and the final has the maximum for the expectation bound to hold.

Remark (SOTA). The SOTA result specifically for Problem 3.3.1 is summarized as follows.

$$\text{Primal-Dual 3-LMP} \xrightarrow{\text{Conversion}} \text{3-approximation}$$

$$\text{Greedy 2-LMP} \longrightarrow \text{2-bipoint rounding} \longrightarrow \text{4-approximation}$$

$$\text{Dual Fitting [Coh+22] } 1.9\ldots 9\text{-LMP} \longrightarrow 1.3\ldots 3\text{-bipoint rounding}^{a} \longrightarrow 2.67\text{-approximation}$$

But we'll see that by changing the problem a bit, like consider squaring the distance in the objective of Problem 3.3.1 (which is the k-mean problem), we can get 9-approximation by Primal-Dual, while the lower path doesn't tell us anything, which is so fragile.

[a] This will return $k + c$ centers, where c is an absolute constant. There's a way to transform this solution back to k centers without loosing any performance.

Note (Derandomized). It's possible to derandomized Algorithm 3.6.

Lecture 8: Local Search for k-Median

We'll now see a completely different algorithm which solve Problem 3.3.1 with $(3+\epsilon)$-approximation ratio by local search. 26 Sep. 10:30

3.3.3 Local Search

The idea is to iteratively improve the current solution. We first see the algorithm.

Algorithm 3.7: k-Median – Local Search

Data: A set of clients $P \subseteq X$, a set of (possible) facilities $Q \subseteq X$, $k \in \mathbb{N}$, width w
Result: A set of opened facilities $Q' \subseteq Q$ with $|Q'| = k$

```
1 Q' ← arbitrary k centers in Q
2 while ∃Q'' s.t. |Q''| = k and cost(Q'') < cost(Q') and |Q'△Q''| ≤ w^a do
3     Q' ← Q''
4 return Q'
```

[a] The symmetric difference $A \triangle B$ is defined as $A \triangle B \coloneqq (A \setminus B) \cup (B \setminus A)$.

Remark (Runtime). In line 2, each iteration in Algorithm 3.7 takes $(n+m)^{O(w)}$ time for $n \coloneqq |P|$ and $m \coloneqq |Q|$. But we have no control of how many iterations Algorithm 3.7 might take since we might decrease the cost by a little each time hence we might fall into exponentially many updates. To solve this, we can ask for

$$\operatorname{cost}(Q'') < (1-\epsilon)\operatorname{cost}(Q')$$

instead to make sure we decrease a reasonable amount each time, which guarantees that we can bound the number of iterations by

$$\log_{\frac{1}{1-\epsilon}}\left(\frac{\operatorname{cost}(\text{starting } Q')}{\mathsf{OPT}}\right).$$

To do the analysis, first note that for any solution Q' output from Algorithm 3.7, we have that there exists no Q'' such that $|Q' \triangle Q''| \leq w$, $|Q''| = k$ and $\operatorname{cost}(Q'') < \operatorname{cost}(Q')$.

Note (Local optimum). We say this Q' is a *local optimum*.

Let $Q^* \subseteq Q$ be the optimal solution, and without loss of generality (by duplicating facilities), assume $Q' \cap Q^* = \varnothing$. We define something called swap.

Notation (Swap). A *swap* $S \subseteq Q' \cup Q^*$ satisfies $|S \cap Q'| = |S \cap Q^*| \leq w/2$.

Note. From local optimality of Q', for any swap S, $\operatorname{cost}(Q') \leq \operatorname{cost}(Q' \triangle S)$.

Now, consider constructing swaps $S_1, \ldots, S_t$ with weights $p_1, \ldots, p_t \in \mathbb{R}^+$ such that $\operatorname{cost}(Q') \leq \operatorname{cost}(Q' \triangle S_i)$ for all i, we have

$$\sum_{i=1}^{t} p_i \cdot (\operatorname{cost}(Q') - \operatorname{cost}(Q' \triangle S_i)) \leq 0. \tag{3.2}$$

Our goal is to show that Equation 3.2 implies $\operatorname{cost}(Q') \leq \alpha \cdot \operatorname{cost}(Q^*)$ for some $\alpha \in \mathbb{R}^+$. To do this, we require the set of swaps to have the following properties.

(a) For all $j \in Q^*$, $\sum_{S_i \ni j} p_i = 1$, and let $p' \coloneqq \max_{j \in Q'} \sum_{S_i \ni j} p_i$.

(b) For all $j \in Q^*$, let $\pi(j) \in Q'$ be the facility closest to j. Then if S_i contains $j \in Q'$, $\pi^{-1}(j) \subseteq S_i$.[3]

The existence of such swaps family is ensured by the following lemma.

[3] In particular, if $|\pi^{-1}(j)| > w/2$, then j is not contained in any swap.

Lemma 3.3.1. There exists a family of swaps $S_1, \dots, S_t$ with weights $p_1, \dots, p_t$ such that $\forall j \in Q^*$, $\sum_{S_i \ni j} p_i = 1$ and if $j \in S_i$, $\pi^{-1}(j) \subseteq S_i$ with $p' = \max_{j \in Q'} \sum_{S_i \ni j} p_i = 1 + 2/w$.

Proof. For all $j \in Q'$, we call j

- *big*: if $\left|\pi^{-1}(j)\right| > w/2$.
- *small*: if $\left|\pi^{-1}(j)\right| \in [1, w/2]$.
- *lonely*: if $\left|\pi^{-1}(j)\right| = 0$.

Then for each small or big j, we create a *group* G_j that contains $\pi^{-1}(j)$, j and $\left|\pi^{-1}(j)\right| - 1$ lonely facilities (denote as $R_j \subseteq Q'$). We see that $|G_j| = 2\left|\pi^{-1}(j)\right|$, and we can ensure each lonely facility belongs to exactly 1 group, i.e., $\exists G_1, \dots G_r$ such that each facility belongs to exactly 1 group. It's now clear that how we should create swaps and their corresponding weight:

(a) For small j, let G_j be a swap with weight 1.

(b) For big j, let $w' \coloneqq \left|\pi^{-1}(j)\right|$, then for any $S \subseteq \pi^{-1}(j)$ and $T \subseteq R_j$ with $|S| = |T| = w/2$, we let $(S \cup T)$ be a swap with weight $1/\left(\binom{w'-1}{w/2-1} \cdot \binom{w'-1}{w/2}\right)$.[a]

Since for every $j^* \in Q^*$, there is only one group containing j^*, to verify $\sum_{S_i \ni j^*} p_i = 1$, we see that

(a) j^* is containing in G_j for j small: In this case, we have one swap, i.e., G_j itself with weight 1.

(b) j^* is containing in G_j for j big: In this case, since every such swap created inside G_j contains j^* and has uniform weight, it sums up to 1.

Finally, we want to show that $p' = \max_{j \in Q'} \sum_{S_i \ni j} p_i = 1 + 2/w$. But this is also easy since given $j \in Q'$, the summation is inside G_j, and in particular, S_i is inside G_j as well. Then

(a) j is small: Only swap is G_j itself with weight 1.

(b) j is big: j can't even be in one swap, hence the sum is 0.

(c) j is lonely: In this case, we have

$$\sum_{S_i \ni j} p_i = \frac{1}{\binom{w'-1}{w/2-1}\binom{w'-1}{w/2}} \cdot \binom{w'}{w/2}\binom{w'-2}{w/2-1} = \frac{w'}{w/2}\frac{w/2}{w'-1} = \frac{w'}{w'-1} \leq 1 + \frac{2}{w}.$$

Taking the maximum, we have $p' = 1 + 2/w$ as desired. ∎

[a] Notice that since j is big, so j can't be in any swap, so we have only $w' - 1$ to choose from.

With Lemma 3.3.1, we're ready to prove the following.

Theorem 3.3.2. Algorithm 3.7 is a $(3 + \epsilon)$-approximation algorithm for arbitrary small $\epsilon > 0$.

Proof. Fix $i \in P$, we analyze how it contributes to the left-hand side of Equation 3.2. Let $j' \in Q'$ and $j^* \in Q^*$ be facilities closest to i, and $d'_i \coloneqq d(i, j')$, $d^*_i = d(i, j^*)$, then for every S_ℓ, we have

(a) $S_\ell \ni j^*$. Then contribution of i to $\operatorname{cost}(Q') - \operatorname{cost}(S_\ell \triangle Q')$ is at least $d'_i - d^*_i$.[a]

(b) $S_\ell \ni j'$. By the second property, either

- $\pi(j^*) \in S_\ell$: this implies $j^* \in S_\ell$, which falls back to the first case.
- $\pi(j^*) \notin S_\ell$: the contribution of i to $\operatorname{cost}(Q') - \operatorname{cost}(S_\ell \triangle Q')$ is at least $d'_i - (2d^*_i + d'_i) = -2d^*_i$.[b]

(c) Otherwise, contribution of i to $\operatorname{cost}(Q') - \operatorname{cost}(S_\ell \triangle Q')$ is at least 0.[c]

In all, we see that the first case has total weight 1 from the first property, while (b)$-$(a) has total weight $\leq p'$, hence Equation 3.2 implies

$$\sum_{i \in P} [(d_i' - d_i^*) \cdot 1 - (2d_i^*) \cdot p'] \leq \sum_{\ell=1}^{t} p_\ell(\operatorname{cost}(Q') - \operatorname{cost}(S_\ell \triangle Q')) \leq 0,$$

which is equivalent to say

$$\operatorname{cost}(Q') - (1 + 2p') \mathsf{OPT} \leq 0,$$

so we get a $(1 + 2p')$-approximation ratio. Furthermore, From Lemma 3.3.1, we have $p' = 1 + 2/w$, hence we can achieve $1 + 2(1 + 2/w) = 3 + 4/w$-approximation ratio. Given $\epsilon > 0$, by setting $w := 4/\epsilon$, we're done. ■

[a] i can go to j^*.
[b] i can go to $\pi(j^*)$.
[c] i can stay with j'.

Lecture 9: Euclidean k-Median

3.4 Euclidean k-Median

28 Sep. 10:30

If we now consider the metric space to be exactly $(\mathbb{R}^\ell, \|\cdot\|_2)$, we get some advantages from the structure of Euclidean metric.

Intuition. The problematic part is the old approach for k-median problem is when i contributing to too many facilities at once. But we'll see that this can't happen if we're considering Euclidean metric, which has some inherent geometric limitation in terms of volume.

Now, let's see the problem formulation.

Problem 3.4.1 (Euclidean k-median). Given a metric space $(X, d) = (\mathbb{R}^\ell, \|\cdot\|_2)$ and $P, Q \subseteq X$ with $k \in \mathbb{N}$, find $Q' \subseteq Q$ with $|Q'| = k$ which minimizes $\sum_{i \in P} \min_{j \in Q'} d(i, j)$.

It's natural to ask that whether we can solve Euclidean k-median like how we solve facility location and k-median. The answer is yes, and in particular, we're going to modify Algorithm 3.1 for facility location to get an α-LMP approximation with $\alpha < 3$, which essentially implies an α-approximation algorithm for k-median using Euclidean metric.

3.4.1 Euclidean Facility Location

Formally, we define the following problem.

Problem 3.4.2 (Euclidean facility location). Given a metric space $(X, d) = (\mathbb{R}^\ell, \|\cdot\|_2)$ and $P, Q \subseteq X$, $f \in \mathbb{R}^+$ where P is the set of clients, Q is the set of (possible) facilities, we want to open $Q' \subseteq Q$ such that it minimizes $\sum_{i \in P} \min_{j \in Q'} d(i, j) + f\,|Q'|$.

As previously seen. Recall the dual of facility location is

$$\begin{aligned}
\max \quad & \sum_i \alpha_i \\
& \alpha_i - \beta_{ij} \leq d(i, j) \quad \forall i, j \\
& \sum_i \beta_{ij} \leq f \qquad\qquad \forall j \\
(D) \quad & \alpha, \beta \geq 0
\end{aligned}$$

and the Algorithm 3.1 uses primal-dual method, where we interpret α_i is the time that i is connected.

Let t_j be the time that j is open in Algorithm 3.1, and the only thing we change is the phase two, i.e., how we trim down the solution. We now see the algorithm, which essentially achieves $\rho := (1 + \delta)$-LMP approximation for $\delta := \sqrt{8/3} \approx 1.633\ldots$.

Algorithm 3.8: Euclidean Facility Location – Primal-Dual

Data: A set of clients $P \subseteq X$, a set of (possible) facilities $Q \subseteq X$, facility cost f
Result: A set of opened facilities $Q' \subseteq Q$

$S \leftarrow \varnothing$, $Q' \leftarrow \varnothing$, $\alpha \leftarrow 0$ // S: connected clients, O: open facilities

while $S \neq P$ **do**
  while *True* **do**
    increase all $\{\alpha_i\}_{i \in P \setminus S}$ by a *unit* **if** *some* $j \in Q \setminus Q'$ *s.t.* $\sum_{i \in P} \beta_{ij} = f$ **then** // j gets tight (open)
      break
    else if *some* $i \in P \setminus S$ *s.t.* $\alpha_i \geq d(i, j)$ **then** // i can connect to $j \in Q'$
      break
  $Q' \leftarrow \{\text{tight facilities}\}$ // Update Q'
  $S \leftarrow \{\text{clients connected to } Q'\}$ // Update S

// Trim down Q'
$G = (Q', E := \{(j, j') : \exists i \in P \text{ such that } d(j, j') \leq \delta \cdot \min(t_j, t_{j'}), j, j' \in Q'\})$
Compute Q'' s.t. $\forall j \in Q'$, either $j \in Q''$ or $\exists j' \in Q''$ s.t. $(j, j') \in E$ // max independent set
return Q''

To do the analysis, as before, let $w(i) \in Q'$ for all i such that $\alpha \geq t_{w(i)}$, i.e., $w(i)$ is the connected witness of i.

Remark. We have the following.

(a) α is dual-feasible.

(b) If $\beta_{ij} > 0$, then $\alpha_i \leq t_j$.

(c) For all i, $\exists w(i) \in Q'$ such that $\alpha_i \geq t_{w(i)}$.

We can do the analysis similarly. Fix a client $i \in P$, then observe that given $S = Q'' \cap \{j : \beta_{ij} > 0\}$, if $\delta = 2$, then $|S| \leq 1$.[4] We see that

(a) If $|S| = 1$, $S = \{j\}$. We see that $\operatorname{conn}(i) \leq d(i, j)$ and $\operatorname{open}(i) = \alpha_i - d(i, j)$, so

$$\operatorname{conn}(i) + \operatorname{open}(i) \leq d(i, j) + (\alpha_i - d(i, j)) \leq \alpha_i.$$

(b) If $|S| = 0$, then $\operatorname{open}(i) = 0$ and either $w(i) \in Q''$, or $j' \in Q''$ such that $(w(i), j') \in E$. In any case, $\operatorname{conn}(i) \leq d(i, j') \leq d(i, w(i)) + d(w(i), j')$, hence

$$\operatorname{conn}(i) + \operatorname{open}(i) \leq d(i, j') \leq \alpha_i + \delta t_{w(i)} \leq (1 + \delta)\alpha_i.$$

Generally, our goal is to prove that for all i,

$$\frac{\operatorname{conn}(i)}{\rho} + \operatorname{open}(i) \leq \alpha_i, \tag{3.3}$$

which implies

$$\frac{\operatorname{conn}}{\rho} + |Q''| f \leq \sum_i \alpha_i,$$

i.e., we get a ρ-LMP approximation algorithm.

[4] Since if both β_j and $\beta_{j'}$ is greater than 0, then $d(j, j') \leq 2\alpha_i \leq 2\min(t_j, t_{j'})$. This means j and j' will have an edge but from the property of max independent set, one of them will not be included.

Note. Specifically, Equation 3.3 is equivalent to

$$\frac{\min_{j\in S} d(i,j)}{\rho} + \sum_{j\in S}(\alpha_i - d(i,j)) \leq \alpha_i.$$

In the case of $\delta = 2$, we see that we can set $\rho \coloneqq 1+\delta = 3$. We see that we get the exactly 3-LMP approximation for $\delta = 2$ case! Notice that in this case, since $|S| \leq 1$ as we noted, Algorithm 3.8 and Algorithm 3.1 are equivalent.

Now, we'll see how can we get advantages by further restricting δ, which utilizes the following.

Remark (k-means magic formulas)**.** There are two extremely useful tricks call *k-means magic formulas* for Euclidean metric related problems. Let $i' = \sum_{j\in S} j/\left|S\right|$. Then

$$\begin{aligned}\sum_{j\in S}\left\|j-i\right\|^2 &= \sum_{j\in S}\left\langle j-i+i'-i', j-i+i'-i'\right\rangle\\ &= \sum_{j\in S}\left(\left\|j-i\right\|^2 + \left\|i'-i\right\|^2 + 2\left\langle j-i', i'-i\right\rangle\right) = \sum_{j\in S}\left\|j-i'\right\|^2 + \left|S\right|\left\|i'-i\right\|^2.\end{aligned}$$

Also,

$$\begin{aligned}\sum_{j,j'\in S}\left\|j-j'\right\|^2 &= \sum_{j,j'\in S}\left\langle j-j'+i'-i', j-j'+i'-i'\right\rangle\\ &= \sum_{j,j'\in S}\left(\left\|j-i'\right\|^2 + \left\|j'-i'\right\|^2 + 2\left\langle j-i', i'-j'\right\rangle\right) = 2\left|S\right|\cdot\sum_{j\in S}\left\|j-i'\right\|^2.\end{aligned}$$

One can actually show that i' (i.e., the geometric mean) is the optimal solution for k-means, and if we choose i rather than i' to be the center, the deviation from OPT is exactly $\left|S\right|\left\|i'-i\right\|^2$. Nevertheless, we have the following.

Lemma 3.4.1. For $\delta \coloneqq \sqrt{8/3}$ and $S = Q'' \cap \{j\colon \beta_{ij} > 0\}$, $|S| \leq 3$.

Proof. From the k-means magic formulas, we have

$$\left|S\right|\alpha_i^2 \geq \sum_{j\in S}\left\|j-i\right\|^2 \geq \frac{1}{2\left|S\right|}\sum_{j,j'\in S}\left\|j-j'\right\|^2 > \frac{(s-1)\delta^2\alpha_i^2}{2},$$

where the last inequality follows from $\left\|j-j'\right\| > \delta\cdot\min(t_j, t_{j'}) \geq \delta\cdot\alpha_i$. Then, we have

$$\left|S\right|\alpha_i^2 > \frac{(s-1)\delta^2\alpha_i^2}{2} \Rightarrow \left|S\right|\left(\frac{\delta^2}{2}-1\right) < \frac{\delta^2}{2} \Rightarrow \left|S\right| < \frac{\delta^2}{\delta^2-2} = 4$$

by plugging in $\delta = \sqrt{8/3}$, hence $|S| \leq 3$ by integrality. ∎

From Lemma 3.4.1, we see that we already handle the case that $|S| = 0$ and $|S| = 1$, so the only cases left are $|S| = 2$ and $|S| = 3$. And by doing so, we obtain the following theorem.

Theorem 3.4.1. Algorithm 3.8 is a $(1+\sqrt{8/3})$-LMP approximation algorithm.

Proof. As said, from Lemma 3.4.1, we only need to consider the case that $|S| = 2$ and $|S| = 3$. If $|S| = 2$, let $S = \{j_1, j_2\}$, then $(\alpha_i - d(i,j_1)) + (\alpha_i - d(i,j_2)) \leq (2-\delta)\alpha_i$. Since $\operatorname{conn}(i) \leq \alpha_i$,

$$\frac{d(i,j^*)}{\rho} + \sum_{j\in S}(\alpha_i - d(i,j)) \leq \alpha_i\left(\frac{1}{\rho}+2-\delta\right) \leq \alpha_i, \tag{3.4}$$

where the last inequality follows from $1/\rho + 2 - \delta \leq 1 \Leftrightarrow 1/\rho \leq \delta - 1$, which is satisfied by $\rho \coloneqq 1+\delta = 1+\sqrt{8/3}$.

If $|S| = 3$, let $S = \{j_1, j_2, j_3\}$. Now, instead of looking at a more complicated geometric structure and try to optimize it, we simply add Equation 3.4 three times for (j_1, j_2), (j_2, j_3) and (j_1, j_3), we have $2\sum_{j\in S}(\alpha_i - d(i,j)) \leq 3(2-\delta)\alpha_i$ hence

$$\frac{d(i,j^*)}{\rho} + \sum_{j\in S}(\alpha_i - d(i,j)) \leq \alpha_i \left(\frac{1}{\rho} + \frac{3(2-\delta)}{2}\right) \leq \alpha_i$$

since $1/\rho + 3(2-\delta)/2 \leq 1 \Leftrightarrow 1/\rho \leq (3\delta - 4)/2$, which is satisfied by $\rho \coloneqq 1 + \delta = 1 + \sqrt{8/3}$ ■

Remark (SOTA). Compare general metric Problem 3.3.1 and Euclidean metric Problem 3.4.1, we have the following.

$$2.41\text{-LMP}^{a} \xrightarrow{\text{Conversion}} 2.41\text{-approximation}$$

$$\text{Euclidean Primal-Dual } 2.63\text{-LMP} \xrightarrow{\text{Conversion}} 2.63\text{-approximation}$$

$$\text{Primal-Dual } 3\text{-LMP} \xrightarrow{\text{Conversion}} 3\text{-approximation}$$

$$\text{Dual Fitting [Coh+22] } 1.9\ldots9\text{-LMP} \longrightarrow 1.3\ldots3\text{-bipoint rounding} \longrightarrow 2.67\text{-approximation}$$

$$\text{Local Search } 3\text{-LMP} \longrightarrow 2\text{-bipoint rounding} \longrightarrow 4\text{-approximation}$$

Noticeably, 2.41-LMP approximation is $1+\sqrt{2}$, which is exactly the threshold behavior in Euclidean metric we're building our intuition upon.

[a] 2.40 is the SOTA.

Note. We assume that ℓ is large throughout. If it's not the case, then actually for all $\epsilon > 0$, there exists a $(1+\epsilon)$-approximation algorithm with running time $2^{2^{O(\ell)}} \cdot \mathsf{poly}(n)$. Hence, if ℓ is small (or constant), we can use this algorithm, otherwise, what we have discussed is better.

Chapter 4

Traveling Salesman Problem

Lecture 10: Spanning Tree

Instead of discussing general network design problems, we focus on traveling salesman problem specifically. And turns out that although this is a good old problem in TCS, but still, lots of improvement is done in the past decade. Turns out, most of the improvement is based on the understanding of spanning tree, specifically, how to sample a good enough random spanning tree. 3 Oct. 10:30

4.1 Spanning Tree

We first look at the definition of a spanning tree.

Definition 4.1.1 (Spanning tree). A *spanning tree* T of a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an induced subgraph of $\mathcal{G}$ which spans $\mathcal{G}$, i.e., $V(T) = \mathcal{V}$ and $E(T) \subseteq \mathcal{E}$.

Remark. A spanning tree of a connected graph $\mathcal{G}$ can also be defined as a maximal set of edges of $\mathcal{G}$ that contains no cycle, or as a minimal set of edges that connect all vertices.

Then, we're interested in the following problem.

Problem 4.1.1 (Minimum spanning tree). Given a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and an edge-weight function $w\colon \mathcal{E} \to \mathbb{R}^+$, find a spanning tree T which minimizes $w(T)$.

There are lots of different algorithms which solve Problem 4.1.1, e.g., Prim's algorithm, Kruskal's Algorithm, etc. in undergraduate algorithm courses. But turns out that by looking at the LP formulation of this problem, we get some non-trivial result.

4.1.1 Spanning Tree Polytope

Denote the variables as $\{x_e\}_{e\in\mathcal{E}}$, where we interpret $x_e = 1$ if e is in the final spanning tree, otherwise if it's 0, then e is not in the final spanning tree.

One natural formulation is

$$
\begin{aligned}
\min \ & \sum_{e\in\mathcal{E}} x_e w(e) \\
& \sum_{e\in\partial S} x_e \geq 1 \quad \forall S \subseteq \mathcal{V} \\
& x \geq 0,
\end{aligned}
$$

where the second constraint is trying to model that for every cut set $S \subseteq \mathcal{V}$, our spanning tree need to include at least one edge from the boundary, i.e., ∂S.

Notation. If $S \subseteq \mathcal{V}$, then we denote $\partial S = E(S, \overline{S})$ be the edges between S and $\overline{S}$.

But turns out that this formulation will give us an integrality gap of 2, since for a cycle graph, just by choosing half of the edges, i.e., $x_e = 1/2$ for all $e \in \mathcal{E}$, the constraints are satisfied while we know we need to include all but one edge to form a valid spanning tree.

Remark. There are ways to strengthen the second constraints by looking at **directed** spanning trees rather than the usual undirected ones to give us an LP which solves Problem 4.1.1 exactly.

We see that the problems arise from the fact that there are not enough edges to span $\mathcal{G}$, so we now require it explicitly in our LP formulation. Furthermore, to ensure there are no cycles, for any $S \subseteq \mathcal{V}$, we again make sure that the total edges we have is less than $|S| - 1$. Then, we have the following spanning tree polytope.

$$
\begin{aligned}
\min \sum_{e \in \mathcal{E}} & x_e w(e) \\
\sum_{e \in \mathcal{E}} & x_e = n - 1 \\
\sum_{e \in E(S)} & x_e \leq |S| - 1 \quad \forall \varnothing \neq S \subsetneq \mathcal{V} \\
x & \geq 0
\end{aligned} \tag{4.1}
$$

where $E(S)$ denotes the set of edges inside S, i.e.,

$$
E(S) \coloneqq \{ e = (u, v) \in \mathcal{E} \colon u, v \in \mathcal{V} \} .
$$

This is not solvable just by throwing this into an LP solver since there are exponentially many constraints! Regardless, we note the following.

Remark (Separation oracle)**.** Given a linear program (P) with $x \in \mathbb{R}^n$ as variables, a *separation oracle* is an algorithm which outputs

- True if x is feasible.
- False with *the violating constraint* if x is not feasible.

And if we have a polynomial time separation oracle, we can solve any LP in polynomial time by using the ellipsoid algorithm.

Now, we just state that there's a separation oracle for the above LP, so we can solve it in polynomial time and get a fractional solution $\{x_e\}_{e \in \mathcal{E}}$. So our next task is to round it into an integral one.

4.1.2 Pipage Rounding

The reason we call Equation 4.1 the *polytope* is that there's a way to transform the any optimal (potentially fractional) solution to this LP can be transformed to an integral one while maintaining the objective value. This can be done via the so-called *pipage rounding* as we'll now see.

Notation (Tight)**.** The set $S \subseteq \mathcal{V}$ is *tight* if $\sum_{e \in E(S, \overline{S})} x_e = |S| - 1$.

Lemma 4.1.1 (Uncrossing)**.** If S and T are tight with $S \cap T \neq \varnothing$, both $S \cup T$ and $S \cap T$ are tight.

Proof. Observe that since S and T are tight and $S \cup T$ and $S \cap T$ are cuts as well (hence satisfy the constraints),

$$
\begin{aligned}
(|S| - 1) + (|T| - 1) &= \sum_{e \in E(S)} x_e + \sum_{e \in E(T)} x_e \\
&\leq \sum_{e \in E(S \cup T)} x_e + \sum_{e \in E(S \cap T)} x_e \leq (|S \cup T| - 1) + (|S \cap T| - 1),
\end{aligned}
$$

with the fact that $|S| + |T| = |S \cup T| + |S \cap T|$,[a] hence everything is equal. ■

[a]Consider every possible edges between $S \setminus T$, $T \setminus S$, $S \cap T$ and $\overline{S \cup T}$.

Finally, we call a tight T *integral* if and only if for all $e \in E(T)$, $x_e \in \{0, 1\}$; and a tight T *fractional* if there exists $e \neq f \in E(T)$ such that x_e and x_f are fractional.[1] We first see the deterministic rounding algorithm.

Algorithm 4.1: Minimum Spanning Tree – Pipage-Rounding

Data: A connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, edge weight $w \colon \mathcal{E} \to \mathbb{R}^+$, solution x of Equation 4.1[a]
Result: A minimum spanning tree T

```
1  while x ∉ ℕ^m do                                          // not integral
2      T ← minimal tight fraction set                        // inclusion-wise minimal^b
3      f, g ← fractional edges                               // f, g ∈ E(T)
4      if w(f) > w(g) then                                   // ensure w(f) ≤ w(g)
5          swap(f, g)
6      while increase x_f and decrease x_g by a unit do      // by solving Equation 4.2
7          if x_f or x_g becomes integral then
8              break
9          else if ∃T' ⊊ T is tight then
10             break
11 T ← Subgraph(𝒢, x)                                        // construct a spanning tree
12 return T
```

[a]By using separation oracle.
[b]i.e., $\nexists T' \subsetneq T$ tight fractional set.

Remark (Implementation detail)**.** There are two non-trivial steps in Algorithm 4.1.

- line 6: This continuous process is done by taking δ from solving the following LP as the total unit we should increase/decrease:

$$
\begin{aligned}
\max\ & \delta \\
& y = x + \delta e_f - \delta e_g \\
& \sum_{e \in E(S)} y - e \leq |S| - 1 \quad \forall S \subseteq \mathcal{V} \\
& 0 \leq y \leq 1,
\end{aligned}
\tag{4.2}
$$

  where e_i is the unit vector with 1 at entry i. Again, this is in the similar form as Equation 4.1, and there's a separation oracle which solves this LP in polynomial-time.

- line 2: Start from the whole vertex set $\mathcal{V}$, and we simply look at f which is none-integral edge and ask can we increase it or not, i.e., we ask the separation oracle for Equation 4.2, and if there's a smaller tight fraction set inside T, $\delta > 0$ strictly, and we just keep searching in this way. We'll see what does this mean exactly in Lemma 4.1.2.

Our goal now is to show that during the pipage rounding, x remains feasible and $\sum_{e \in \mathcal{E}} x_e w(e)$ will not increase.

We first show that $\sum_{e \in \mathcal{E}} x_e w(e)$ will not increase. This is because from our design, $\sum_{e \in \mathcal{E}} x_e$ remains unchanged, while we increase x_f while decrease x_g for $w(f) \leq w(g)$, hence the total cost for the spanning tree decreases.

To show x remains feasible, first note that the non-tight sets are handled (captured) in line 9, as for tight sets, we have Lemma 4.1.2.

Lemma 4.1.2. All tight sets remain tight after running line 6.

[1]We can equivalently require only one x_e being fractional, but since T is tight, there'll another $f \neq e$ such that x_f is fractional as well.

Proof. The only way for a tight set becomes over-tight is when we increase x_f in line 6, an already tight set U becomes over-tight. But if this is the case and U is violated, then $U \ni f$ and $U \not\ni g$ and U is tight, we have $U \cap T$ is tight from Lemma 4.1.1, contradicting the minimality of T ↯ ∎

Remark. From the poof, we can now find minimal T by increasing a fractional x_f: if some set U is not violated, then $T \cap U$ is tight, so we just keep nesting and get the minimal one.

Now, it remains to show Algorithm 4.1 terminates in polynomial time.

Lemma 4.1.3. Algorithm 4.1 in a polynomial time algorithm.

Proof. Observe that

(a) line 7 can only happen m times: at most m edges can be fractional at first, and after one becomes integral, it remains integral.

(b) line 9 can only happen n times: at most n nodes can be in T at first, and when line 9 is triggered, the size of T decreases by at least 1 and never goes up.

In all, we see that Algorithm 4.1 is a polynomial time algorithm. ∎

Note. Notice that in line 9, we require $T' \subsetneq T$, and if it's triggered, in the next iteration when choosing T in line 2, we'll need to choose a strictly smaller T compare to the last iteration[a] in order to make Lemma 4.1.3 valid.

[a]This is guaranteed by Lemma 4.1.2 since we know the only we change is x_f and x_g, and if some new T' can become tight, it has non-empty intersection with T and hence as the remark, we can find such a T'.

We see that this implies the following.

Theorem 4.1.1. Algorithm 4.1 solves Problem 4.1.1 exactly in polynomial time.

Proof. Firstly, Algorithm 4.1 is a polynomial time algorithm from Lemma 4.1.3. Also, since Equation 4.1 is an LP-relaxation of Problem 4.1.1 while we know that ∎

And indeed, we have a randomized version of Algorithm 4.1.

Algorithm 4.2: Minimum Spanning Tree – Randomized Pipage-Rounding

Data: A connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, edge weight $w\colon \mathcal{E} \to \mathbb{R}^+$, solution x of Equation 4.1[a]
Result: A minimum spanning tree T

while $x \notin \mathbb{N}^m$ **do** // not integral
  $T \leftarrow$ minimal tight fraction set // inclusion-wise minimal[b]
  $f, g \leftarrow$ fractional edges // $f, g \in E(T)$
  if $w(f) > w(g)$ **then** // ensure $w(f) \leq w(g)$
    `swap`(f, g)
  $a \leftarrow \max_a x_f \leftarrow x_f + a, x_g \leftarrow x_g - a$ remain feasible // $a > 0$
  $b \leftarrow \max_b x_f \leftarrow x_f - b, x_g \leftarrow x_g + b$ remain feasible // $b > 0$
  if `rand`$((0,1)) < \frac{b}{a+b}$ **then** // w.p. $\frac{b}{a+b}$
    $x_f \leftarrow x_f + a$, $x_g \leftarrow x_g - a$
  else // w.p. $\frac{a}{a+b}$
    $x_f \leftarrow x_f - b$, $x_g \leftarrow x_g + b$
$T \leftarrow$`Subgraph`$(\mathcal{G}, x)$ // construct a spanning tree
return T

[a]Again, by using separation oracle.
[b]i.e., $\nexists T' \subsetneq T$ tight fractional set.

As in the deterministic version, the same proof can show that x is feasible, and the number of iteration will be less than $m \cdot n$, hence it's a polynomial time algorithm. Remarkably, we have the following.

Theorem 4.1.2. Algorithm 4.2 solves Problem 4.1.1 exactly.

Proof. To show that the cost is good enough, note that in one iteration, $\mathbb{E}\left[x^{\text{end}}\right] = x^{\text{start}}$, then

$$\mathbb{E}\left[x^{\text{final}}\right] = x^{\text{LP}},$$

hence any possible x^{final} satisfies

$$\sum_{e\in\mathcal{E}} x_e^{\text{final}} w(e) = \sum_{e\in\mathcal{E}} x_e^{\text{LP}} w(e),$$

hence we get a 1-approximation algorithm, i.e., Algorithm 4.2 solves Problem 4.1.1 exactly. ■

From Algorithm 4.2, x^{final} can be interpreted as the distribution of spanning trees, i.e., we have

$$\mathbb{E}\left[x^{\text{final}}\right] = x^{\text{LP}} \Leftrightarrow \forall e \in \mathcal{E}, \Pr(e \in T) = x_e^{\text{LP}},$$

where the probability depends on the randomness introduce in Algorithm 4.2, i.e., x_e^{final}. So, from now on, when we say we sample a spanning tree from x, what we mean is to construct a spanning tree w.r.t. the solution x to the spanning tree polytope using Algorithm 4.2.

4.2 Negative Correlation

One of the reasons why we're interested in Algorithm 4.2 is because it produces a negative correlated distribution, which leads to a strong concentration behavior, i.e., we have control on what kind of spanning tree we're going to get. Firstly, if x_e^{final} are independent, then

$$\mathbb{E}\left[\prod_{e\in S} x_e^{\text{final}}\right] = \Pr(S \subseteq T) = \prod_{e\in S} \Pr(e \in T) = \prod_{e\in S} x_e^{\text{LP}}.$$

But since we know that x_e^{final} are not independent for sure since they depend on a sequence of steps executed by Algorithm 4.2, it's non-trivial to analyze. We now see the main result in this section.

Theorem 4.2.1 (Negative correlation)**.** For all $S \subseteq \mathcal{E}$,

$$\mathbb{E}\left[\prod_{e\in S} x_e^{\text{final}}\right] = \Pr(S \subseteq T) \leq \prod_{e\in S} \Pr(e \in T) = \prod_{e\in S} x_e^{\text{LP}}.$$

Proof. Let y^i be x after i^{th} iteration maintained by Algorithm 4.2, it's sufficient to show

$$\mathbb{E}\left[\prod_{e\in S} y_e^{i+1} \mid y^i\right] \leq \prod_{e\in S} y_e^i$$

since if this holds, say Algorithm 4.2 runs M iterations in total, then

$$\mathbb{E}\left[\prod_{e\in S} x_e^{\text{final}}\right] = \mathbb{E}\left[\prod_{e\in S} y_e^M\right] = \mathbb{E}\left[\prod_{e\in S} y_e^M \mid y^{M-1}\right] \leq \prod_{e\in S} y_e^{M-1},$$

any by taking expectation again iteratively, we obtain the desired result down to $\prod_{e\in S} y_e^0$. Now, consider that in the i^{th} iteration of Algorithm 4.2, for f, g picked in line 3:

(i) $f, g \notin S$: trivially holds.

(ii) $f \in S$, $g \notin S$:[a] we have $\mathbb{E}\left[\prod_{e\in S} y_e^{i+1} \mid y^i\right] = \prod_{e\in S\setminus\{f\}} y_e^i \cdot \mathbb{E}\left[y_f^{i+1} \mid y^i\right] = \prod_{e\in S} y_e^i$ where $\mathbb{E}\left[y_f^{i+1} \mid y^i\right] = y_f^i$ is the designed from Algorithm 4.2.

(iii) $f, g \in S$. Suffices to compare $\mathbb{E}\left[y_f^{i+1} \cdot y_g^{i+1} \mid y^i\right]$ and $y_f^i \cdot y_g^i$, and the goal is to show $\leq$.

(a) $\mathbb{E}\left[(y_f^{i+1}+y_g^{i+1})^2 \mid y^i\right] = (y_f^i+y_g^i)^2$ since $y_f^{i+1}+y_g^{i+1} = y_f^i+y_g^i$ almost surely.

(b) $\mathbb{E}\left[(y_f^{i+1}-y_g^{i+1})^2 \mid y^i\right] \geq (y_f^i+y_g^i)^2$ since the variance of any random variable is non-negative.

We see that by subtracting them, we have $\mathbb{E}\left[y_f^{i+1}\cdot y_g^{i+1} \mid y^i\right] \leq y_f^i \cdot y_g^i$ as desired.

In all cases, the hypothesis for i holds, hence the theorem is proved. ■

[a] And also $g \in S$ and $f \notin S$, since they're symmetric.

Lecture 11: Asymmetric TSP

5 Oct. 10:30

As previously seen. We have shown that given any feasible x, there's a distribution of spanning tree T such that

(a) For all $e \in \mathcal{E}$, $\Pr(e \in T) = x_e$

(b) For all $S \subseteq \mathcal{E}$, $\Pr(S \subseteq T) \leq \prod_{e\in S} x_e$ from Theorem 4.2.1.

From these, we can deduce the following.

Theorem 4.2.2. For all $S \subseteq \mathcal{E}$ and $\gamma \geq 1$,

$$\Pr\left(\lvert S\cap T\rvert \geq \gamma \sum_{e\in S} x_e\right) \leq \left(\frac{e}{\gamma}\right)^{\gamma \sum_{e\in S} x_e}.$$

Proof. This follows directly from the same proof of Chernoff bound. Assume we have k random variables $X_1, \ldots, X_k \in \{0,1\}$ with $X = \sum_{i=1}^k X_i$ and $\mu = \mathbb{E}[X]$. Then,

$$\Pr(X \geq \gamma\mu) = \Pr(e^{tX} \geq e^{t\gamma\mu}) \leq \frac{\mathbb{E}\left[\prod_{i=1}^k e^{tX_i}\right]}{e^{t\gamma\mu}}$$

where the inequality follows from Markov's inequality. If X_i are independent, we can move the expectation inside the product, but if we don't, we directly apply Theorem 4.2.1 to get the same result, so we can proceed as usual. ■

4.3 Asymmetric Traveling Salesman Problem

Now we can talk about the asymmetric traveling salesman problem. Before we state the problem, we first look at one important definition.

Definition 4.3.1 (Tour). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a *tour* $(a_0, \ldots, a_k)$ where $a_i \in \mathcal{V}$ satisfies $a_0 = a_k$, $(a_i, a_{i+1}) \in \mathcal{E}$ and visited all the vertices, i.e., $\{a_i\}_{i=0}^k = \mathcal{V}$.

Problem 4.3.1 (Asymmetric TSP). Given a complete bidirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a distance function $d\colon \mathcal{E} \to \mathbb{R}^+$ satisfying the *directed* triangle inequality.[a] *Asymmetric TSP* asks to find a tour $(a_0, \ldots, a_k)$ which minimizes $\sum_{i=0}^{k-1} d(a_{i-1}, a_i)$.

[a] Compare to the regular triangle inequality, now the order matters, i.e., for all $a, b, c \in \mathcal{V}$, $d(a,c) \leq d(a,b) + d(b,c)$.

Remark. An equivalent (but seemingly more general) formulation of Problem 4.3.1 is to remove the complete graph restriction and also the directed triangle inequality property of d. But they're still equivalent since given this general problem, we can convert back to Problem 4.3.1 by setting

$$d'(u, v) = \min d(u, v).$$

Note (SOTA). The approximation ratio of Problem 4.3.1 is improved as follows.

$$\begin{tikzcd}
\lg n \arrow[r, squiggly, "1982\sim 2009"] & c\lg n \arrow[r, "2009\sim 2010"] & O(\log n/\log\log n) \arrow[d, "2010\sim 2018"] & \\
& & 5500 \arrow[r, "2018\sim 2020"] & 22
\end{tikzcd}$$

where in 2009, $c \in (0, 1)$.

4.3.1 Asymmetric TSP Polytope

We now try to solve Problem 4.3.1. The idea is simple, given $T \subseteq \mathcal{E}$ for T being a multiset, we want T to satisfy

(a) T is connected (in undirected sense)

(b) T is Eulerian: $\deg^+_T(v) = \deg^-_T(v)$ for all $v \in \mathcal{V}$[2]

which allow us to potentially construct a valid tour by shortcut some repetitions if there's any. We then have the following LP formulation, which is the so-called asymmetric TSP polytope. Denote our variables as $\{x_e\}_{e\in\mathcal{E}}$, then

$$\begin{aligned}
\min \ & \sum_{e\in\mathcal{E}} x_e d(e) & & \\
& \sum_{e\in\partial^+ S} x_e \geq 1 & & \forall \varnothing \neq S \subsetneq \mathcal{V} \\
& \sum_{e\in\partial^+\{v\}} x_e = \sum_{e\in\partial^-\{v\}} x_e = 1 & & \forall v \in \mathcal{V} \\
& x \geq 0 & &
\end{aligned} \tag{4.3}$$

where $\partial^+ S := \{(u, v) \in \mathcal{E} \mid u \in S, v \notin S\}$ and vice versa, and $\partial S := \partial^+ S \cup \partial^- S$. Now, to solve this LP, the idea is to maintain Eulerianity while gradually being more connected. This can be done via cycle cover LP.

As previously seen. Recall that $C \subseteq \mathcal{E}$ is a cycle cover if c is disjoint union of directed cycles and v is in exactly one cycle.

Now, we have the following.

Lemma 4.3.1. There is a cycle cover C such that $\sum_{e\in C} d(e) \leq \mathsf{OPT}_{\mathrm{LP}}$.

To prove Lemma 4.3.1, we need to have some understanding about the perfect matching polytope. This is not that well-known since matching problem can be solved in many ways.

Remark (Perfect matching polytope). Suppose we have an unweighted bipartite graph $\mathcal{G} = (A \sqcup B, \mathcal{E})$ and a weight function $w\colon \mathcal{E} \to \mathbb{R}^+$. We want to find a perfecting matching with minimum cost.

[2] We use $\deg^+$ to denote the out degree, while $\deg^-$ to denote the in degree.

This can be modeled by the following LP.

$$
\begin{aligned}
\min \sum_{e\in\mathcal{E}} x_e w(e) & \\
\sum_{e\in E(u, v-u)} x_e = 1 \quad & u \in A \sqcup B \\
x \geq 0. &
\end{aligned}
$$

This LP is exact in the sense that for any feasible x, there exists a perfect matching (integral solution) M such that $\sum_{e\in M} w(e) \leq \sum_{e\in\mathcal{E}} x_e w(e)$.

Now we can prove Lemma 4.3.1.

Proof of Lemma 4.3.1. We simply construct a complete bipartite graph with vertex set $\mathcal{V}_{\text{out}} \sqcup \mathcal{V}_{\text{in}}$ such that the $\mathcal{V}_{\text{out}} = \mathcal{V}_{\text{in}} = \mathcal{V}$ with the edge weight being $x_{(a,b)}$ for a in the left-hand side while b in the right-hand side.

Observe that in B, every vertex has x value exactly 1, hence from perfect matching polytope, we know that there exists a perfect matching M in B with cost $(\sum_{e\in M} w(e))$ less the LP cost $(\sum_{e\in\mathcal{E}} x_e w(e))$, with the fact that M corresponds to a cycle cover in the original graph by considering picking $(a, b) \in M$ the directed edge $(a, b) \in \mathcal{E}$, so we're done. ■

Then, we have the following algorithm.

Algorithm 4.3: Asymmetric TSP – Cycle Covered

Data: A connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, distance function $d\colon \mathcal{E} \to \mathbb{R}^+$
Result: A tour T

$\mathcal{C} \leftarrow$ minimum cycle cover of $\mathcal{G}$
$\mathcal{V}' \leftarrow \varnothing$
for $C \in \mathcal{C}$ **do**
  $x \leftarrow$`rand`(C) // Choose one representative
  $\mathcal{V}' \leftarrow \mathcal{V}' \cup \{x\}$
$T \leftarrow$`ATSP`$(\mathcal{G}[\mathcal{V}'], d)$ // tour among representatives
return $T \leftarrow$`Stitch`$(\mathcal{C}, T)$

`Stitch`$(\mathcal{C}, T)$:
  for $C \in \mathcal{C}$ **do**
    $T \leftarrow T \cup C$ // Connects T with C
  return T

Theorem 4.3.1. Algorithm 4.3 is a $\lg n$-approximation algorithm.

Proof. We simply observe that for every recursive call of solving cycle cover LP, since we don't have self-loops, so the number of vertices, $\mathcal{V}'$, constructed in Algorithm 4.3 will decrease by a factor of 2 since every cycle need at least two vertices, so the total number of recursive calls will be at most $\lg n$. From the fact that in each recursive call, the cost will be at most the cost of the original LP solution for the entire graph from Lemma 4.3.1,[a] so by adding the cost up (i.e., stitching the tour together), the total cost will be at most $\lg n \cdot \mathsf{OPT}$, proving the result. ■

[a] Recall that we're recursively solving for subgraph of $\mathcal{G}$.

Remark (Repetition). Observing that in Algorithm 4.3, our construction might not return a valid Eulerian tour. But by triangle inequality, we can always skip some vertices when we encounter already visited vertices, so we're still fine.

4.3.2 Reducing to Thin Tree

Now let's see more sophisticated approach to Problem 4.3.1 where we first make sure T is connected, and try to make it Eulerian afterwards. One problematic case is that when there's one $S \subseteq \mathcal{V}$ such that T has lots of edges in ∂S, then since we want to ensure $\deg_T^+(v) = \deg_T^-(v)$ for all $v \in \mathcal{V}$, by summing up for all v, we'll need to balance this out by (potentially) adding lots of edges on top of T to make it Eulerian.

Definition 4.3.2 ((α, β)-thin)**.** A tree $T \subseteq \mathcal{E}$ is (α, β)-*thin* if $\sum_{e\in T} d(e) \leq \alpha\,\mathsf{OPT}$ and $|T \cap \partial S| \leq \beta \sum_{e\in\partial S} x_e$ for all $\varnothing \neq S \subsetneq \mathcal{V}$.

Let's first see a lemma.

Lemma 4.3.2. We can construct an $(\alpha + 2\beta)$-approximation tour from an (α, β)-thin tree.

Proof. Suppose we have an (α, β)-thin tree T, we want to find a *multi-subgraph* $f\colon \mathcal{E} \to \{0\} \cup \mathbb{N}$ such that

(a) $f(e) \geq 1$ for all $e \in T$

(b) $\sum_{e\in\partial^+\{v\}} f(e) = \sum_{e\in\partial^-\{v\}} f(e)$ for all $v \in \mathcal{V}$.

We can still define an LP as follows.

$$\begin{aligned} \min \sum_{e\in\mathcal{E}} & f(e)d(e) \\ & f(e) \geq 1 && \forall e \in T \\ & \sum_{e\in\partial^+ S} f(e) \geq \left|T \cap \partial^- S\right| && \forall \varnothing \neq S \subsetneq \mathcal{V} \\ & f \geq 0. \end{aligned}$$

Claim. The above LP is exact in the sense that if we have an LP solution f, we can get a tour of cost $\sum_{e\in\mathcal{E}} f(e)d(e)$.

Proof. This is just like max-flow min-cut theorem. ⊛

Now, let y be

$$y_e = \begin{cases} 1 + 2\beta x_e, & \text{if } e \in T; \\ 2\beta x_e, & \text{if } e \notin T, \end{cases}$$

and the goal is to show y_e is feasible to the LP. But the only non-trivial constraints we need to check is $\sum_{e\in\partial^+ S} y_e \geq |T \cap \partial^- S|$. This follows from

$$\sum_{e\in\partial^+ S} y_e \geq 2\beta \sum_{e\in\partial^+ S} x_e = \beta \sum_{e\in\partial S} x_e \geq |T \cap \partial S| \geq \left|T \cap \partial^- S\right|.$$

Hence, y is a feasible solution of the LP, so we get a tour with cost $\sum_{e\in\mathcal{E}} y_e d(e)$, which is just

$$\sum_{e\in\mathcal{E}} y_e d(e) = \sum_{e\in T} d(e) + 2\beta \cdot \mathsf{OPT}_{\mathrm{LP}} \leq (\alpha + 2\beta)\,\mathsf{OPT}_{\mathrm{LP}}$$

since T is itself (α, β)-thin, which proves the result. ■

We see that Problem 4.3.1 boils down to finding an (α, β)-thin tree. To do this, we'll show that by randomly sampling a spanning tree, it'll be a thin tree with high probability. But the argument is non-trivial, and turns out that the number of small cuts (approximate min-cuts) is important, so we now look into this.

4.3.3 Number of Small Cuts

Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a weight function $x \colon \mathcal{E} \to \mathbb{R}^+$, denote λ to be the minimum edge-connectivity, i.e.,

$$\lambda := \min_{\varnothing \neq S \subsetneq \mathcal{V}} \sum_{e \in \partial S} x(e),$$

we want to ask how many S achieves the value λ, i.e., how many edge min-cuts are there? It's a well-known fact that the number of the min-cuts are n^2 (or $\binom{n}{2}$ to be exact), which is tight. Now, we're interested in approximate min-cuts, or α-cuts: given $\alpha \in \mathbb{N}$, we ask that how many α-cuts S are there where an α-mincuts is defined as cuts which achieves $\sum_{e \in \partial S} x(e) \leq \alpha \cdot \lambda$? The following theorem answers this.

Theorem 4.3.2. For all $\alpha \in \mathbb{N}$, there are at most $2n^{2\alpha}$ α-mincuts.

Lecture 12: ATSP with Random Spanning Tree

Let's prove Theorem 4.3.2. 10 Oct. 10:30

Proof of Theorem 4.3.2. We first see a randomized algorithm which solves the α-mincuts problem.

Algorithm 4.4: Small α-Mincuts – Karger's Algorithm [Kar93]

Data: A connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a weight function $x \colon \mathcal{E} \to \mathbb{R}^+$, α
Result: An α-mincut S

while $|\mathcal{V}| > 2\alpha$ **do**
  $e \leftarrow$`rand(`$\mathcal{E}$, x_e`)` // Sample w.p. x_e
  `contract(`$\mathcal{G}$, e`)` // new x is the sum of multi-edges' x
$S \leftarrow$`rand-subset(`$\mathcal{V}$`)` // $|\mathcal{V}| = 2\alpha$
$S \leftarrow$`uncontract(`S`)`
return S

Now, if S is an α-mincut, we're interested in the probability of S is outputted from Algorithm 4.4.

Remark. Observe that if $e = (u, v)$ is contracted where $u \in S$ while $v \notin S$, then S will definitely not be outputted.

Denote $\Pr(S \text{ survives when } |\mathcal{V}| = i) =: P_i$, then if S survived until $|\mathcal{V}| = i$, we see that S is still an α-mincut, and in the current $\mathcal{G}$, by considering a single vertex as a potential mincut, we have

$$\frac{\sum_{e \in \partial S} x_e}{\alpha} \leq \lambda \leq \mathbb{E}_{v \in \mathcal{V}}\left[\sum_{e \in \partial\{v\}} x(e)\right] = \frac{2}{i} \sum_{e \in \mathcal{E}} x(e) \Rightarrow P_i = 1 - \frac{\sum_{e \in \partial S} x(e)}{\sum_{e \in \mathcal{E}} x(e)} \geq 1 - \frac{2\alpha}{i} = \frac{i - 2\alpha}{i}$$

since $P_i = 1 - \Pr(S \text{ does not survive when } |\mathcal{V}| = i)$, while the latter event happens when one of the boundary edges in ∂S is picked to be contracted. We then see that

$$\begin{aligned}
\Pr(S \text{ is outputted}) &\geq P_n \cdot P_{n-1} \cdot \cdots \cdot P_{2\alpha+1} \cdot \frac{1}{2^{2\alpha}} \\
&= \left(\frac{n - 2\alpha}{n}\right)\left(\frac{n - 1 - 2\alpha}{n - 1}\right) \cdots \left(\frac{1}{2\alpha + 1}\right) \cdot 2^{-2\alpha} \\
&= \frac{(2\alpha)!}{n(n-1)\ldots(n-2\alpha+1)} \cdot 2^{-2\alpha} \\
&\geq \frac{1}{(2n)^{2\alpha}},
\end{aligned}$$

then the result follows because the probability should sum to 1. ■

4.3.4 Random Thin Spanning Tree

Now, we're ready to show that a randomly sampled spanning tree will be a thin tree with high probability. Recall the spanning tree polytope, with our ATSP polytope for Problem 4.3.1. Then given an optimal x of Equation 4.3, define $y_{uv} = x_{uv} + x_{vu}$ and $z = \frac{n-1}{n} y$ for Equation 4.1.[3]

Remark. z is feasible for the spanning tree polytope.

Proof. Since we have scaled down y, hence $\sum_{e\in\mathcal{E}} z_e = n-1$. While for the second constraint, all $\varnothing \neq S \subsetneq \mathcal{V}$, we have

$$\sum_{e\in E(S)} x_e = \sum_{v\in S}\sum_{e\in\partial^+\{v\}} x(e) - \sum_{e\in E(S,\overline{S})} x(e) \leq |S| - 1,$$

so z is feasible since the above sum doesn't care about direction as well, and we even decrease it a bit by down-scaling. ⊛

Now, we can sample a random spanning tree T' by using the randomized pipage rounding on z. Specifically, we have the following algorithm.

Algorithm 4.5: Thin Spanning Tree – Randomized Pipage-Rounding

Data: A connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, weight $w\colon \mathcal{E} \to \mathbb{R}^+$, solution z of Equation 4.1
Result: A minimum spanning tree T

```
T' ← Randomized-Pipage-Rounding(G, w, z)
T ← ∅
for e = {u, v} ∈ T' do                  // Using the cheaper edge between (u, v) and (v, u)
    if w((u, v)) ≤ w((v, u)) then
        T ← T ∪ {(u, v)}
    else
        T ← T ∪ {(v, u)}
return T
```

We now show that T obtained from Algorithm 4.5 is indeed a thin tree.

Lemma 4.3.3. T output from Algorithm 4.5 satisfies $\alpha = 2$ for (α, β)-thinness with probability at least $1/2$.

Proof. Since

$$\mathbb{E}\left[d(T)\right] \leq \frac{n-1}{n}\,\mathsf{OPT}_{\mathrm{LP}}$$

since we have $z_{uv}\cdot\min(d(u,v), d(v,u))$ v.s. $x_{uv}d(u,v) + x_{uv}d(v,u)$. We then see that with probability at least $1/2$ by simply using Markov's inequality, hence the first condition of thin tree is satisfied with probability at least $1/2$. ■

Lemma 4.3.4. T output from Algorithm 4.5 satisfies $\beta = 12\log n/\log\log n$ for (α,β)-thinness with probability at least $1 - n^{-1}$.

Proof. Given any β, we want to bound the probability that there's one $\varnothing \neq S \subsetneq \mathcal{V}$ which violates the condition. To do this, let

$$C_i := \left\{ S \subseteq \mathcal{V} \mid z(\partial S) \in [i\lambda, (i+1)\lambda) \right\},$$

where we have

$$\lambda = \min_{\varnothing\neq S\subseteq\mathcal{V}} z(\partial S) = 2\cdot\frac{n-1}{n} \geq 1.$$

Now, recall that Algorithm 4.5 uses Algorithm 4.2, hence $e \in T'$ satisfies the negative correlation, which essentially allows us to prove Theorem 4.2.2. Specifically, we have

[3]Notice that the summation over y is exactly n, but we want the sum to be $n-1$, hence we scale it down.

(a) for all $e \in \mathcal{E}$, $\Pr(e \in T') \coloneqq z_e$,

(b) for all $\varnothing \neq E \subseteq \mathcal{E}$, $\Pr(E \subseteq T') \leq \prod_{e\in\mathcal{E}} z_e$,

and with this, the Chernoff bound-like concentration states that

$$\Pr(|T' \cap S| > \beta \cdot z(\partial S)) \leq \left(\frac{e}{\beta}\right)^{\beta\cdot z(\partial S)}.$$

Then, we have

$$\begin{aligned}
\Pr(\exists S\colon\ |T \cap \partial S| > \beta z(\partial S)) &\leq \sum_{i=1}^{\infty} \Pr(\exists S \in C_i\colon\ |T \cap \partial S| > \beta z(\partial S)) \\
&\leq \sum_{i=1}^{\infty} (2n)^{2(i+1)} \cdot \left(\frac{e}{\beta}\right)^{\beta\cdot i\lambda} \\
&\leq \sum_{i=1}^{\infty} (2n)^{2(i+1)} \cdot \left(\frac{e}{\beta}\right)^{\beta\cdot i},
\end{aligned}$$

where we drop λ since $\lambda \geq 1$ as we have shown, which is an even-weaker bound. From the concentration bound, we see that by taking $\beta = c \cdot \log n / \log\log n$ for some constant c,

$$\begin{aligned}
\left(\frac{e}{\beta}\right)^{\beta\cdot i} &= \left(\frac{e\log\log n}{c\log n}\right)^{\frac{c\log n}{\log\log n}\cdot i} \\
&= \exp\left(\frac{c\cdot i\cdot \log n}{\log\log n}(1 + \log\log\log n) - c - \log\log n\right) \\
&\leq \exp\left(\frac{c\cdot i\cdot \log n}{\log\log n}\left(-\frac{\log\log n}{2}\right)\right) \\
&= \exp\left(-\frac{ci}{2}\log n\right) \\
&= n^{-6i},
\end{aligned}$$

where the inequality holds when n is large, and the last equality is obtained from letting $c \coloneqq 12$. Then, we see that

$$\Pr(\exists S\colon\ |T \cap \partial S| > \beta z(\partial S)) \leq \sum_{i=1}^{\infty} (2n)^{2(i+1)} \cdot \left(\frac{e}{\beta}\right)^{\beta\cdot i} \leq \sum_{i=1}^{\infty} (2n)^{2(i+1)} \cdot n^{-6i} \leq \frac{1}{n},$$

as desired. ■

Theorem 4.3.3. T output from Algorithm 4.5 is a $(2, 12\log n/\log\log n)$-thin tree with probability at least $1/2 - n^{-1}$.

Proof. Combining Lemma 4.3.3 and Lemma 4.3.4 and using a union bound argument, we have the desired result. ■

In all, we have the following.

Algorithm 4.6: Asymmetric TSP – Randomized Construction

Data: A connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, weight $w\colon \mathcal{E} \to \mathbb{R}^+$, solution z of Equation 4.1
Result: A tour T

1. $T' \leftarrow$ `Thin-Spanning-Tree`$(\mathcal{G}, w, z)$
2. $T \leftarrow$ `Thin-Tree-to-Tour`(T')
3. **return** T

We finally have the following.

Theorem 4.3.4. Algorithm 4.6 is an $O(\log n / \log\log n)$-approximation algorithm with probability at least $1/2 - n^{-1}$.

Proof. By combining Theorem 4.3.3 and Lemma 4.3.2, we will obtain a $(2 + 24\log n/\log\log n)$-approximation tour, proving the result. ■

4.4 Symmetric Traveling Salesman Problem

Let's now look at a simpler version of Problem 4.3.1.

Problem 4.4.1 (Symmetric TSP). Given a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a distance function $d \colon \mathcal{E} \to \mathbb{R}^+$ satisfying the triangle inequality. *Symmetric TSP* asks to find a tour $(a_0, \ldots, a_k)$ which minimizes $\sum_{i=0}^{k-1} d(a_{i-1}, a_i)$.

4.4.1 Christofides-Serdyuko Algorithm

We first see a simple heuristic algorithm achieves 1.5-approximation ratio of Problem 4.4.1 due to Christofides [Chr76] and Serdyukov [Ser78], discovered independently. Remarkably, this simple heuristic algorithm achieves nearly the best approximation ratio we know for more than 40 years, and the SOTA result achieves $(1.5 - 10^{-36})$-approximation ratio [KKG21].

Algorithm 4.7: Symmetric TSP – Christofides-Serdyuko Algorithm [Chr76; Ser78]

Data: A connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a distance function $d \colon \mathcal{E} \to \mathbb{R}^+$
Result: A tour T

```
1 T ←MST(𝒢, d)                          // Compute a minimum spanning tree
2 O ← {v ∈ T: deg(v) is odd}
3 M ←Min-Matching(O, d)                 // Compute a minimum matching
4 T ← T ∪ M
5 return T
```

Remark. We see that line 2 and line 3 solves the degree problem we have.

Proof. Explicitly, since we want a T to be (connected and) Eulerian, given a spanning tree, the only problematic part is the odd degree vertices, and hence we can just match them to solve the problem. ⊛

Clearly, Algorithm 4.7 runs in polynomial time, and we're interested in bounding the approximation ratio.

Theorem 4.4.1. Algorithm 4.7 is a 1.5-approximation algorithm.

Proof. Denote any optimal solution of Problem 4.4.1 by T^*, which is an optimal tour. Then we simply observe that $d(T') \leq d(T^*)$ and $d(M) \leq d(T^*)/2$, and we have

$$d(T) = d(T') + d(M) \leq 1.5 \cdot d(T^*),$$

proving the claim. ■

However, we get more insight by looking at the LP relaxation of Problem 4.4.1, and in fact, the recent improvement is based on looking into the corresponding LP formulation.

4.4.2 Symmetric TSP Polytope

We now analyze the approximation ratio via LP formulation of Algorithm 4.7. Indeed, since there are no directions now, we may follow the same strategy of how we solve Problem 4.3.1, i.e., we first define the symmetric TSP polytope via a simple reduction from the asymmetric TSP polytope.

$$\begin{aligned} \min \ & \sum_{e\in\mathcal{E}} x_e d(e) \\ & \sum_{e\in\partial S} x_e \geq 2 \qquad \forall \varnothing \neq S \subsetneq \mathcal{V} \\ & \sum_{e\in\partial\{v\}} x_e = 2 \qquad \forall v \in \mathcal{V} \\ & x \geq 0 \end{aligned} \tag{4.4}$$

Let x be this LP optimal solution, we now try to build a tour based on the two-step procedure as in Algorithm 4.7:

(a) finding a spanning tree,

(b) fix it to be in the symmetric TSP polytope by finding a matching of odd degrees vertices.

To do the analysis, observe that $x \cdot \frac{n-1}{n}$ is in the spanning tree polytope as in the case of Problem 4.3.1, which implies if we can find a minimum spanning tree T, then $d(T) \leq \sum_{e\in\mathcal{E}} d(x)x_e$.

Note that it's not enough to just get T, we still need a matching M. Let O be the set of odd-degree vertices w.r.t. T as defined in line 2. Notice that we may assume $|O|$ is even, we then define the following.

Definition 4.4.1 (O-join)**.** Given $O \subseteq \mathcal{V}$ and $|O|$ even, $M \subseteq \mathcal{E}$ is *O-join* if $\deg_M(v)$ is odd when $v \in O$, and $\deg_M(v)$ is even if $v \notin O$.

With this, we define the so-called *O-join LP*.

$$\begin{aligned} \min \ & \sum_{e\in\mathcal{E}} y_e d(e) \\ & y(\partial S) \geq 1 \quad \forall S \text{ s.t. } |S \cap O| \text{ odd;} \\ & y \geq 0. \end{aligned} \tag{4.5}$$

Lemma 4.4.1. The O-join LP is exact, i.e., if y is feasible, then there exists an O-join M such that $d(M) \leq \sum_{e\in\mathcal{E}} d(e)y_e$.

We omit the proof here, but if we believe Lemma 4.4.1 is true, then we have the following.

Theorem 4.4.2. Algorithm 4.7 is a 1.5-approximation algorithm using LP relaxation analysis.

Proof. Since both spanning tree polytope and the O-join LP are valid LP relaxation of line 1 and line 3, respectively, by denoting T and M obtained from solving these two LPs respectively, from the above discussion, we know

$$d(T) \leq \frac{n-1}{n} \times \mathsf{OPT}_{\mathrm{LP}}$$

for the spanning tree polytope LP and

$$d(M) \leq \frac{1}{2} \times \mathsf{OPT}_{\mathrm{LP}}$$

for the O-join LP, combining these we have

$$d(T \cup M) = d(T) + d(M) \leq 1.5 \cdot \mathsf{OPT}_{\mathrm{LP}}$$

for the STSP polytope, as desired. ■

Lecture 13: Toward Next Step: Magic Spanning Tree Distribution

12 Oct. 10:30

4.5 Beyond the $3/2$ Barrier for STSP

In this section, we'll see some ideas of the recent breakthrough on symmetric TSP, which breaks the $3/2$-approximation barrier by a tiny absolutely constant [KKG21].

> **As previously seen.** If we have a solution x in the STSP polytope, then $(n-1)/n \times x$ is in the spanning tree polytope and $x/2$ is in the O-join LP.

The advantage we'll get is that, indeed, dividing x by 2 is wasteful, or more explicitly, we want to find the set of odd degree vertices O w.r.t. T which defines the O-join LP with the cost less than $(1/2-\delta)x$ for some absolutely constant $\delta > 0$.

> **Intuition.** The slackness comes from the difference between all cuts and O-odd cuts.

But observe that O depends entirely on the choice of T, and hence our goal now is to sample a spanning tree T (with the corresponding $O = O^{(T)}$) such that the feasible solution $y^{(T)}$ for the corresponding $O^{(T)}$-join LP satisfies

(a) $\mathbb{E}_T\left[e \in T\right] = (n-1)/n \cdot x_e$,

(b) $\mathbb{E}_T\left[y_e^{(T)}\right] \leq (1/2-\delta)x_e$.

If this is the case, we see that

$$\mathbb{E}\left[\text{cost of TSP tour}\right] \leq \left(\frac{3}{2}-\delta\right) \cdot \mathsf{OPT}_{\mathrm{LP}},$$

which breaks the $3/2$ barrier for symmetric TSP.

4.5.1 Strong Assumptions

Surely, we'll not look into the most general setting, instead, we'll make some strong assumptions to help us get intuitions. Specifically, we assume that there exists a tiny constant $\epsilon \in (0, 0.01)$ such that

1. $x_e \leq \epsilon$,
2. $\sum_{e\in\partial S} x(e) \geq 2+\epsilon$ for all non-singleton cut S.[4]

The second assumption is a huge assumption, but if we have this, we see that $x/(2+\epsilon)$ satisfies all O-join LP constraints except for the singleton cut S, and we're going to fix this. To do so, we first define a useful terminology.

> **Notation** (Even). Given a tree T, the edge $e=(u,v)$ is *even* if both $\deg_T(u)$ and $\deg_T(v)$ are even.

Then, we see that if we can sample a T such that

(a) $\mathbb{E}_T\left[e \in T\right] = (n-1)/n \cdot x_e$,

(b) $\mathbb{E}_T\left[e \text{ is even}\right] \geq \delta$,

then we'll have

$$y_e^{(T)} = \begin{cases} x_e/(2+\epsilon), & \text{if } e \text{ is even;} \\ x_e/2, & \text{otherwise,} \end{cases}$$

which is always in the O-join polytope since if e is even, then the O-join LP constraint is irrelevant in this case, and hence

$$\mathbb{E}\left[y_e^{(T)}\right] \leq \delta\frac{x_e}{2+\epsilon} + (1-\delta)\frac{x_e}{2} < \left(\frac{1}{2}-\frac{\delta\epsilon}{4}\right)x_e,$$

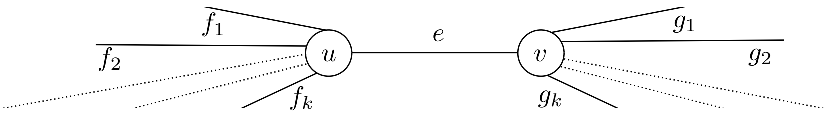
we again get the slackness we want. Now, the goal is to find such a spanning tree distribution, which is the tricky part.

[4] This means $2 \leq |S| \leq n-2$.

4.5.2 Characterization of Spanning Tree Distribution

To find such a distribution, we start by characterizing some properties which are necessary for any distribution satisfies the two conditions above. In particular, we care more about the second condition, i.e., $\mathbb{E}_{\mathcal{T}}[e \text{ is even}] \geq \delta$, since the first one is quite easy to satisfy.

Fix a $(u, v) \in \mathcal{E}$, we want to make sure that $\Pr_{\mathcal{T}}(\deg_T(u) \text{ and } \deg_T(v) \text{ are even}) \geq \delta$. Let f_i and g_i be the edges incident to u and v, respectively, and by abusing the notations, we also let f_i and g_i be the indicator variables indicating whether f_i appears in the tree T or not.



Note. Notice that $\deg_{\mathcal{G}}(u) = \deg_{\mathcal{G}}(v)$ since in symmetric TSP, $\mathcal{G}$ is complete.

Now, define $d_u := \deg_T(u) = e + f_1 + \cdots + f_k$ and $d_v := \deg_T(v) = e + g_1 + \cdots + g_k$, we know that $d_u, d_v \geq 1$, and to satisfy the first condition, i.e., $\mathbb{E}_{\mathcal{T}}[e \in T] = (n-1)/n \cdot x_e$, we have

$$\mathbb{E}[d_u] = \mathbb{E}[e + f_1 + \cdots + f_k] = \mathbb{E}[d_v] = \mathbb{E}[e + g_1 + \cdots + g_k] = \frac{n-1}{n} \cdot 2$$

since in the symmetric TSP polytope requires that $\sum_{e \in \partial\{v\}} x_e = 2$. We're now interested in characterizing the probability density function of d_u, which we now know it's value is at least 1 and the mean is around 2.

Log-Coincavity

Let's first introduce some definition.

Definition 4.5.1 (Log-concave). Let a be an integer-valued random variable, then the distribution of a is *log-concave*[a] if

$$\Pr(a = i) \geq \sqrt{\Pr(a = i-1)\Pr(a = i+1)}.$$

[a]If we take the log on both sides, we'll get a concave function.

We see that Definition 4.5.1 can be equivalently characterize as

$$\Pr(a = i+1) \leq \Pr(a = i) \cdot \left(\frac{\Pr(a = i)}{\Pr(a = i-1)}\right).$$

Intuition (Unimodality). If a distribution is log-concave, going from $a = i$ to $a = i-1$, the probability decrease by a factor of α, then when considering $i+1$, it'll decrease even faster. This property is called the *unimodality*.

Example (Binomial distribution). The *binomial distribution* is log-concave. Furthermore, it's almost an *if and only if* condition in our case.

Proof. If we look at d_u, we can think of it as the distribution of getting heads when flipping biased coins independently with different probability each time. ⊛

With this interpretation, the distribution of d_u is like

(a) minimum value 1,

(b) mean 2,

(c) essentially binomial,

hence it shouldn't behave too crazily. We can now prove the following.

Claim. $\Pr(d_u = 2) \geq 1/5$.

Proof. Let $\Pr(d_u = 2) =: b < 1/5$, and let $a := \Pr(d_u = 1)$, then $a \geq (1 - 1/5)/2 = 2/5$ since otherwise $\Pr(d_u \geq 3) > 2/5$ and $\mathbb{E}[d_u] > 2$, contradiction. Then, we see that the probability (ratio) gap between a and b is approximately $1/2$, from log-concavity, the probability sum will just not sum up to 1, contradiction again.[a] ⊛

[a] If a is large, then b need to be small, this large gap with log-concavity will make sure the same argument follows.

Also, by the similar argument, we have the following.

Claim. $\Pr(d_u \geq 2) \geq 1/2$.

Conditioning

Here, we'll see that some good properties hold after conditioning. Explicitly, let E be the event that $d_u + d_v = 4$, by the same argument as above where we now know that $d_u + d_v$ has minimum value of 2 with mean nearly 4, we have $\Pr(E) \geq 1/10$. Now, we're going to consider the probability of $\Pr(d_u = i)$ conditioning on E, i.e., $\Pr(d_u = i \mid E)$.

Remark. After conditioning, it's still log-concave.

We see that if $\Pr(d_u = 2 \mid E) \geq 10\delta$, then we're done since

$$\Pr(d_u = d_v = 2) \geq \Pr(E) \cdot \Pr(d_u = 2 \mid E) \geq \frac{1}{10} \cdot 10\delta = \delta.$$

4.5.3 Toward a Contradiction

What we want is *almost* the case, since if $\Pr(d_u = 2 \mid E) \leq 10\delta$, from $10\sigma < 1/3$ and the log-concavity and the only value d_u and d_v can take is $1, 2$, and 3, we know that 2 can't be a mode. Hence, we see that

- 1 is the mode: $\Pr(d_u = 3 \mid E) \leq 10\delta$,
- 3 is the mode: $\Pr(d_u = 1 \mid E) \leq 10\delta$,

since we're now assuming $\Pr(d_u = 2 \mid E) \leq 10\delta$. We see that since d_u and d_v are symmetric, we may just assume

- $\Pr(d_u \leq 2 \mid E) \leq 20\delta$,
- $\Pr(d_v \geq 2 \mid E) \leq 20\delta$.

We're now going to show that this leads to a contradiction, and hence what we want is indeed the case, i.e., $\Pr(d_u = 2 \mid E) \geq 10\delta$, leading to $\Pr(d_u = d_v = 2) \geq \delta$.

Stochastic Dominance

Intuitively, stochastic dominance states that if $a + b$ is big, then a is big. From the above discussion with this intuition, we have

- $\Pr(d_u \leq 2 \mid d_u + d_v \geq 4) \leq 20\delta$,
- $\Pr(d_v \geq 2 \mid d_u + d_v \leq 3) \leq 20\delta$.

Note. From these two bound, we're almost saying something like $d_u + d_v$ is positively correlated with d_u, and is also positively correlated with d_v.

With this intuition, we see that

$$\Pr(d_u \leq 2 \text{ and } d_v \geq 2) \leq 20\delta$$

since the original two probability bounds' events are disjoint, so only one will happen. Then, we have

$$\begin{aligned}\mathbb{E}\left[d_u \cdot d_v\right] &= \mathbb{E}\left[d_u \cdot d_v \mid d_v = 1\right] \Pr(d_v = 1) + \mathbb{E}\left[d_u \cdot d_v \mid d_v \geq 2\right] \Pr(d_v \geq 2) \\ &\geq \Pr(d_v = 1) + \mathbb{E}\left[3 \cdot d_v \mid d_v \geq 2\right] \Pr(d_v \geq 2) - O(\delta) \\ &\geq 5 - O(\delta),\end{aligned} \tag{4.6}$$

where in the first inequality, in the first term, we drop $\mathbb{E}\left[d_u \cdot d_v \mid d_v = 1\right]$ naively since this is always greater than 1, and for the second term, we almost have $d_u \geq 3$ given $d_v \geq 2$ from the following intuition.

> **Intuition.** If we first consider $\delta = 0$, then given $d_v \geq 2$, we know that from the second bound, $d_u + d_v \nleq 3$, i.e., $d_u + d_v \geq 4$. From the first bound, this further implies $d_u \nleq 2$, i.e., $d_u \geq 3$.

Also, for the second inequality, since $\Pr(d_v = 1) \leq 1/2$ and

$$\mathbb{E}\left[d_v\right] = \Pr(d_v = 1) + \mathbb{E}\left[d_v \mid d_v \geq 2\right] \Pr(d_v \geq 2) \approx 2,$$

the whole sum is minimized when $\Pr(d_v = 1) = 1/2$, and we get $5 - O(\delta)$.

> **Remark.** We see that although d_u, d_v is nearly 2 as we know, but their product is at least 5.

Negative Correlation

Consider calculating $\mathbb{E}\left[d_u \cdot d_v\right]$ as follows,

$$\begin{aligned}\mathbb{E}\left[d_u \cdot d_v\right] &= \mathbb{E}\left[\left(e + \sum_i f_i\right)\left(e + \sum_i g_i\right)\right] \\ &= \mathbb{E}\left[e^2\right] + \sum_i \left(\mathbb{E}\left[e\right]\mathbb{E}\left[f_i\right] + \mathbb{E}\left[e\right]\mathbb{E}\left[g_i\right]\right) + \sum_{ij} \mathbb{E}\left[f_i\right]\mathbb{E}\left[g_j\right] \\ &= \mathbb{E}\left[e\right] + \sum_i \left(\mathbb{E}\left[e\right]\mathbb{E}\left[f_i\right] + \mathbb{E}\left[e\right]\mathbb{E}\left[g_i\right]\right) + \sum_{ij} \mathbb{E}\left[f_i\right]\mathbb{E}\left[g_j\right] \\ &\leq \epsilon + \mathbb{E}\left[d_u\right]\mathbb{E}\left[d_v\right] \\ &\leq 4 + \epsilon,\end{aligned} \tag{4.7}$$

where we use the assumption that $x_e \leq \epsilon$. We see that Equation 4.6 and Equation 4.7 gives us a contradiction.

In all, as discussed in subsection 4.5.1, we arrive at the following theorem.

> **Theorem 4.5.1.** Under the assumptions, any spanning tree distribution satisfies log-concavity, conditioning property, stochastic dominance property, and also negative correlation property, the 3/2 barrier of symmetric TSP can be broken.

4.5.4 Magic Spanning Tree Distribution

To see the complexity about the general cases, we now state the spanning tree distribution we'll need in the general case. There are actually three relevant distributions: given $x \in [0, 1]^{|\mathcal{E}|}$ in the spanning tree polytope, let a distribution μ of a random spanning tree T, i.e., let $\mu(T) = \Pr(T \text{ is sampled})$.

Strongly Rayleigh Distribution

> **Definition.** Given a distribution $\mu(T)$ and $p(z_1, \ldots, z_m) := \sum_T \mu(T) \prod_{e \in T} z_e$.
>
> > **Definition 4.5.2** (Real-stable)**.** We say p is *real-stable* if $\mu(T) \in \mathbb{R}$ for all T, and for all $z_1, \ldots, z_m \in \mathbb{C}$ such that $\operatorname{Im}(z_i) > 0$ for all i, we have $p(z_1, \ldots, z_m) \neq 0$.

Definition 4.5.3 (Strongly Rayleigh distribution). If μ induces a real-stable p, then μ is a *strongly Rayleigh distribution*.

Remark (Closure). If $p(z_1, \ldots, z_n)$ is real-stable, so are

(a) $p(z_1, z_1, z_3, \ldots, z_n)$,

(b) $p(a, z_2, z_3, \ldots, z_n)$ for all $a \in \mathbb{R}$,

(c) $\partial p/\partial z_1$,

(d) etc.

Turns out that there are lots of distributions are strongly Rayleigh, and it's indeed a very good choice in our case since it satisfies all log-concavity, conditioning property, stochastic dominance property, and also negative correlation.

Max-Entropy Distribution

Nevertheless, among all strongly Rayleigh distributions, we want to choose a most random one. This suggests we look into the notion of entropy. Denote ST be the set of all spanning trees, consider the following *maximum entropy LP*:

$$\begin{aligned} \max & \sum_{T \in \mathrm{ST}} \mu(T) \log \frac{1}{\mu(T)} & & & \min & \sum_{T \in \mathrm{ST}} \mu(T) \log \mu(T) & \\ & \sum_{T \in \mathrm{ST}} \mu(T) = 1 & & \Leftrightarrow & & \sum_{T \in \mathrm{ST}} \mu(T) = 1 & \\ & \sum_{T \ni e} \mu(T) = x_e & \forall e \in \mathcal{E} & & & \sum_{T \ni e} \mu(T) = x_e & \forall e \in \mathcal{E} \\ & \mu \geq 0. & & & & \mu \geq 0. & \end{aligned} \tag{4.8}$$

Solving this induces the following.

Definition 4.5.4 (Max-entropy distribution). The optimal solution μ for the maximum entropy LP is the *max-entropy distribution*.

Observe that the maximum entropy LP has exponentially many variables, but at least this is a convex program since $x \log x$ is a convex function, and we can indeed approximately solve it in polynomial time with only polynomially many $T \in \mathrm{ST}$ has non-zero probability, i.e., the size of the support of μ is in polynomial.

λ-Uniform Distribution

Finally, we have the following.

Definition 4.5.5 (λ-uniform distribution). A distribution $\mu(T)$ is *λ-uniform* if there exists a $\lambda \in (\mathbb{R}^+ \cup \{0\})^n$ such that $\mu(T) = \prod_{e \in T} \lambda_e / M$ for some M.

Note (Uniform distribution). When $\lambda_e = 1$ for all $e \in \mathcal{E}$, then we just have a uniform distribution over ST.

Now, we're going to see how these three distributions relate to each other. A technical lemma is the following.

Proposition 4.5.1. A max-entropy distribution is always λ-uniform for some λ.

Proof. We can take the Lagrangian dual of the maximum entropy LP, the optimality condition (i.e., KKT condition) will give us the result. ■

More interestingly, we have the following.

Theorem 4.5.2 (Matrix tree theorem). A λ-uniform distribution μ for ST, then the induced p is real-stable, i.e., μ is strongly Rayleigh.

Combining Proposition 4.5.1 and Theorem 4.5.2, we have the following.

Corollary 4.5.1. A max-entropy distribution is strongly Rayleigh.

We see that it's enough to find a max-entropy distribution, and as noted before, this can be done via solving the maximum entropy LP in polynomial time!

Chapter 5

Semidefinite Programming and Lasserre Hierarchy

Lecture 14: Semidefinite Programming

In this chapter, we'll talk about the Lasserre hierarchy (equivalently Sum of Squares), a good reference is Rothvoß's lecture notes [Rot13]. 19 Oct. 10:30

5.1 Semidefinite Programming

To start with, we first introduce semidefinite programming and first develop some useful tools related to this.

5.1.1 Positive Semidefinite Matrix

In this section, an $n \times n$ matrix A is usually symmetric with real entries. In such a case, we have the following theorem.

As previously seen (Spectrum theorem)**.** Given an $n \times n$ real and symmetric matrix A,

(a) There exists n real eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$.

(b) There exists n eigenvectors $v_1, \ldots, v_n$ which form an orthonormal basis.

This implies we can write $A = \sum_{i=1}^{n} \lambda_i v_i v_i^\top$.[a]

[a]Conversely, if $A = \sum_{i=1}^{n} \lambda_i v_i v_i^\top$, then λ_i will be the eigenvalues with v_i being the corresponding eigenvector.

We now introduce the notion of positive semidefinite matrices, which is the building block of semidefinite programmings.

Definition 5.1.1 (Positive semidefinite)**.** A matrix A is *positive semidefinite (PSD)*, denote as $A \succeq 0$, if for all $x \in \mathbb{R}^n$,

$$x^\top A x = \sum_{ij} x_i A_{ij} x_j \geq 0.$$

Notation. The set of real and symmetric matrices is denoted as $\mathbb{S}^n$, and the set of PSD matrices is denoted as $\mathbb{S}^n_+$.

An equivalent characterization is given by the following.

Lemma 5.1.1. $A \succeq 0$ if and only if all eigenvalues of A is non-negative.

Proof. If $\lambda_n < 0$, we have $v_n^\top A v_n = \lambda_n < 0$. On the other hand, for all $x = \sum_{i=1}^n \lambda_i v_i$, we know that $x^\top A x = \sum_i \alpha_i^2 \lambda_i \geq 0$. ■

Example. Covariance matrix, identity matrix are PSD. Also, given any V, $VV^\top$ is PSD as well.

Proof. For $VV^\top$, we see that for all $x \in \mathbb{R}^n$, we have $x^\top (VV^\top) x = \left\langle V^\top x, V^\top x \right\rangle \geq 0$. ⊛

We see that there is a deeper connection between a PSD matrix and the form of $VV^\top$ for some V, which indeed form another equivalent characterization of PSD matrices.

Lemma 5.1.2. A matrix X is PSD if and only if $X = VV^\top$ for some $V \in \mathbb{R}^{n\times k}$.

Proof. We already see that $VV^\top$ is PSD. Now, given $X \succeq 0$, we can write $X = \sum_{i=1}^n \lambda_i v_i v_i^\top = VV^\top$ where

$$V \coloneqq \operatorname{diag}\left(\left\{\sqrt{\lambda_1}, \dots, \sqrt{\lambda_n}\right\}\right) \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}.$$

■

Remark. Given any two PSD A, B and $\alpha, \beta \geq 0$, $\alpha A + \beta B \succeq 0$.

5.1.2 Semidefinite Programming

Recall that the LP with variables $x \in \mathbb{R}^n$ is in the form of

$$\begin{aligned} \max\ & \langle c, x\rangle \\ & \langle a_i, x\rangle \leq b_i \quad i = 1, \dots, m \\ & x \geq 0, \end{aligned}$$

with input $c, a_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$. We can generalize LP to a much border class of optimization problem by making vectors as matrices, which leads to the so-called semidefinite programming.

Definition 5.1.2 (Semidefinite programming). The *semidefinite programming (SDP)* with variables $X \in \mathbb{S}^n$ is in the form of

$$\begin{aligned} \max\ & \langle C, X\rangle \\ & \langle A_i, X\rangle \leq b_i \quad i = 1, \dots, m \\ & X \succeq 0, \end{aligned}$$

with input $C, A_i \in \mathbb{S}^n$ and $b_i \in \mathbb{R}$.

Remark. We define the inner product between $A, B \in \mathbb{S}^n$ as

$$\langle A, B\rangle \coloneqq \sum_{i,j\in[n]} A_{ij} B_{ij} = \operatorname{tr}(AB).$$

To see that SDP is a generalization of LP, we have the following.

Lemma 5.1.3. SDP captures LP.

Proof. Given an instance of LP, i.e., given input $c, a_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$, then consider defining $C \coloneqq \operatorname{diag}(c)$, $A_i \coloneqq \operatorname{diag}(a_i)$. Then the corresponding SDP is exactly equal to the given LP. ■

Unlike LP, where we can solve it exactly in polynomial time. Here, there's some pathological instances which cause the solution of an SDP exponential to the input size.[1] Nevertheless, we have the following.

Theorem 5.1.1. Most of SDPs can be solved in polynomial time.

[1] This is the so-called bit-complexity problem.

5.1.3 Max Cut

We one can imagine, SDP is usually regarded as a continuous optimization problem, and we now see one application of SDP in approximation algorithm on a combinatorial optimization problem.

Problem 5.1.1 (Max cut). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, find a cut $S \subseteq \mathcal{V}$ which maximizes $|\partial S|$.

Remark. There are various ways of achieving 1/2-approximation for max cut, including greedy, local-search and also LP approaches. But we can prove that 1/2 is tight if we only use the above method, i.e., we can't even improve this approximation ratio a bit.

Let $[n] \coloneqq \mathcal{V}$ given that $n \coloneqq |\mathcal{V}|$, and denote the variables being x_i for $i = 1, \dots, n$ be 1 if $i \in S$, -1 if $i \notin S$. Then the following programming captures max cut when optimizing over $x_i \in \mathbb{R}$:

$$
\begin{aligned}
\max \ & \sum_{(i,j)\in\mathcal{E}} (1 - x_i x_j)/2 \\
& x_i^2 = 1 \qquad\qquad \forall i \in [n].
\end{aligned}
$$

Remark. This is a quadratic programming.

To solve this, we relax $x_i \in \mathbb{R}$ to $u_i \in \mathbb{R}^k$, we have

$$
\begin{aligned}
\max \ & \sum_{(i,j)\in\mathcal{E}} (1 - \langle u_i, u_j \rangle)/2 \\
& \|u_i\|_2^2 = 1 \qquad\qquad \forall i \in [n].
\end{aligned}
$$

Now, from Lemma 5.1.2, we see that $X \succeq 0 \Leftrightarrow X = VV^\top$ for some V. This suggests that the above relaxed programming is a SDP with $V = \begin{bmatrix} u_1^\top & u_2^\top & \dots & u_n^\top \end{bmatrix}$ for $u_i \in \mathbb{R}^k$ such that

$$
\begin{aligned}
\max \ & \sum_{(i,j)\in\mathcal{E}} (1 - X_{ij})/2 \quad \forall i \in [n] \\
& X_{ii} = 1 \\
& X \succeq 0
\end{aligned}
\tag{5.1}
$$

where we let $X_{ij} = \langle u_i, u_j \rangle$. Since this is a relaxation for max cut, we know that $\mathsf{OPT}_{\mathrm{SDP}} \geq \mathsf{OPT}$.

Just like how we do LP relaxation, we first solve Equation 5.1 to get X, and round the solution back to get ± 1 values for x_i to obtain a feasible solution for max cut. A naive rounding algorithm is the following.

Algorithm 5.1: Max Cut – Randomized Rounding

Data: A connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a solution X for Equation 5.1
Result: A cut $S \subseteq \mathcal{V}$

$V \leftarrow$`Eigendecomposition(`X`)` // $X = VV^\top$
$\begin{bmatrix} u_1^\top & u_2^\top & \dots & u_n^\top \end{bmatrix} \leftarrow V$
$g \leftarrow$`rand(`S^{n-1}`)` `// Choose a random direction`
$S \leftarrow \{i \in \mathcal{V} \colon \langle g, u_i \rangle \geq 0\}$ `// Choose one side`
return S

We see that given $(i, j) \in \mathcal{E}$, denote the contribution to the SDP as S_{ij}, and we have

$$
S_{ij} = \frac{1 - \langle u_i, u_j \rangle}{2}.
$$

On the other hand, the expected contribution from (i, j) to Algorithm 5.1, denoted as P_{ij}, is

$$
P_{ij} \coloneqq \Pr((i, j) \text{ is cut edge}) = \Pr(i, j \text{ is separated by } g).
$$

Now, the only thing we need to do is to find the ratio between P_{ij} and S_{ij}. And indeed, this ratio is proved by Goemans-Williamson [GW95].

Notation (Goemans-Williamson constant). The *Goemans-Williamson constant* α_{GW} is defined as

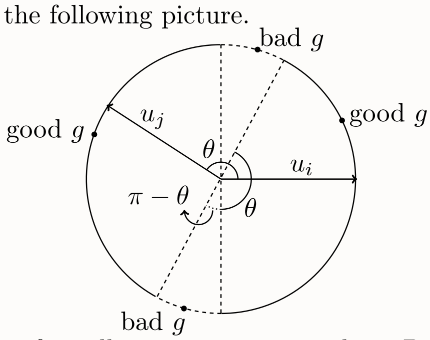
$$\alpha_{\mathrm{GW}} := \frac{2}{\pi} \min_{0 \leq \theta \leq \pi} \frac{\theta}{1 - \cos \theta}.$$

Remark. $\alpha_{\mathrm{GW}} \approx 0.878\ldots$. Moreover, we can instead write α_{GW} as

$$\alpha_{\mathrm{GW}} = \frac{2}{\pi} \min_{-1 \leq \alpha \leq 1} \frac{\arccos(a)}{1 - a}.$$

Lemma 5.1.4. For all $(i, j) \in \mathcal{E}$, $P_{ij} \geq \alpha_{\mathrm{GW}} \cdot S_{ij}$.

Proof. This can be seen from the following picture.



We're interested in what choice of g will not separate u_i and u_j. By drawing lines orthogonal to u_i and u_j, we divide the plane into above four regions. We see that if g lies in the dotted region, then the two inner products $\langle g, u_i \rangle$ and $\langle g, u_j \rangle$ will have the same sign, hence they're not separated.

Now, since $P_{ij} = 2\theta / 2\pi = \theta / \pi$, and recall that $S_{ij} = (1 - \langle u_i, u_j \rangle)/2$, so the worst configuration producing the smallest ratio between P_{ij} and S_{ij} is

$$\frac{P_{ij}}{S_{ij}} \geq \min_{a \in [-1,1]} \frac{\arccos(a)/\pi}{(1 - a)/2} = \alpha_{\mathrm{GW}},$$

where $a := \langle u_i, u_j \rangle$. This proves the result. ∎

Theorem 5.1.2 ([GW95]). Algorithm 5.1 is an α_{GW}-approximation algorithm in expectation.

Proof. Given S outputted from Algorithm 5.1, we see that

$$\frac{\mathbb{E}\left[|S|\right]}{\mathsf{OPT}_{\mathrm{SDP}}} = \frac{\sum_{(i,j) \in \mathcal{E}} P_{ij}}{\sum_{(i,j) \in \mathcal{E}} S_{ij}} \geq \alpha_{\mathrm{GW}}$$

from Lemma 5.1.4, which proves the result. ∎

Lecture 15: Lasserre Hierarchy and Max Cut

5.2 Lasserre Hierarchy

24 Oct. 10:30

The Lasserre hierarchy is a systematic procedure to strengthen a relaxation for an optimization problem by adding additional variables and SDP constraints. In the last years this hierarchy moved into the focus of researchers in approximation algorithms as they obtain relaxations have provably nice properties.

5.2.1 Local Distributions

Firstly, recall max cut and its IP formulation and the SDP relaxation. We're interested in whether there's something *between* the IP and this SDP relaxation?

$$\begin{aligned} &\max \frac{1}{4}\sum_{(i,j)\in\mathcal{E}} \|u_i - u_j\|_2^2 \\ \text{(SDP)}\quad & \|u_i\|_2^2 = 1 \qquad \forall i\in[n], \end{aligned} \quad \to \cdots \to \quad \begin{aligned} &\max \frac{1}{4}\sum_{(i,j)\in\mathcal{E}} (x_i - x_j)^2 \\ \text{(IP)}\quad & x_i \in \{\pm 1\} \qquad \forall i\in[n]. \end{aligned}$$

The upshot is that the SDP solutions are kind of telling us the second moment information. In order to see this, instead of optimizing over ± 1, we now optimize over $\{0, 1\}$.

$$\begin{aligned} &\max \sum_{(i,j)\in\mathcal{E}} \|u_i - u_j\|_2^2 \\ & \|u_\varnothing\|_2^2 = 1 \qquad \forall i\in[n] \\ \text{(SDP)}\quad & \langle u_i, u_\varnothing\rangle = \|u_i\|_2^2 \qquad \forall i\in[n] \end{aligned} \quad \to \cdots \to \quad \begin{aligned} &\max \sum_{(i,j)\in\mathcal{E}} (x_i - x_j)^2 \\ \text{(IP)}\quad & x_i \in \{0, 1\} \qquad \forall i\in[n]. \end{aligned}$$

Now, we ask the following question.

Problem. If $u_\varnothing$, $\{u_i\}_{i\in[n]}$ are feasible to $\{0,1\}$-SDP, what does it tell us about the integral solutions?

Answer. If $\{u_i\}_{i\in[n]}$, $u_\varnothing$ is 1-dimensional, then this solution $\{u_i\}_{i\in[n]}, u_\varnothing$ encodes a cut $S\subseteq\mathcal{V}$ such that

$$S = \{i\in[n] \mid u_i = 1\},$$

in which case we can get

$$\langle u_i, u_j\rangle = \mathbb{1}(i, j\in S)$$

for all $i, j\in[n]$. Now, if $\{u_i\}_{i\in[n]}$, $u_\varnothing$ is **not** 1-dimensional, then we hope that we can view the solution $\{u_i\}_{i\in[n]}, u_\varnothing$ is encoding a *distribution* $\mathcal{D}$ over cuts $S\subseteq\mathcal{V}$, in which case, we can think of

$$\langle u_i, u_j\rangle = \mathbb{E}_{S\sim\mathcal{D}}\left[\mathbb{1}(i, j\in S)\right]$$

for all $i, j\in[n]$, i.e., a covariance matrix. But sadly, this is not true in this exact form since a PSD matrix doesn't always stand for a covariance matrix of some distribution over $\{0,1\}$-valued assignments. ⊛

To get at least some versions of what we want, we first introduce a special kind of distribution called 2-local distribution.

Definition 5.2.1 (2-local distribution). A 2-*local distribution* is a set of distributions consisting of

- $\widetilde{P}_i$: distribution over $\{0,1\}$-assignments for $X_i = \mathbb{1}(i\in S)$ for all $i\in[n]$;
- $\widetilde{P}_{ij}$: distribution over $\{0,1\}$-assignments for (X_i, X_j) for all $(i,j)\in[n]\times[n]$,

which satisfies the 2-local consistency.

Definition 5.2.2 (2-local consistency). The set of distributions $\widetilde{P}_i$ and $\widetilde{P}_{ij}$ is 2-*local consistent* if for all $i, j\in[n]$ and for all $\theta\in\{0,1\}$,

$$\widetilde{P}_i(X_i = \theta) = \sum_{\theta'\in\{0,1\}} \widetilde{P}_{ij}(X_i = \theta, X_j = \theta') = \widetilde{P}_{ij}(X_i = \theta).$$

Example. If $\widetilde{P}_i(X_i = \theta) = 1$ and $\widetilde{P}_{ij}(X_i = \theta) = 0$, then this set is not a 2-local distribution.

Now, consider the $\{0,1\}$-SDP with local probabilities. The 2-local variables are $\{\widetilde{P}_i\}_{i\in[n]} \cup \{\widetilde{P}_{ij}\}_{i,j\in[n]}$

with $\{v_i\}_{i\in[n]}$ and $v_\varnothing$. Then the SDP for max cut is defined as

$$\begin{aligned} \max \sum_{(i,j)\in\mathcal{E}} & \|u_i - u_j\|_2^2 \\ \langle u_i, u_\varnothing\rangle &= \widetilde{P}_i(X_i = 1) && \forall i \in [n] \\ \langle u_i, u_j\rangle &= \widetilde{P}_{ij}(X_i = X_j = 1) && \forall i, j \in [n] \end{aligned}$$

Remark. Technically, we should also introduce another distribution $\widetilde{P}_\varnothing(X_\varnothing) = 1$, and $\widetilde{P}_{ij}$ is defined for all $(i,j) \in ([n] \cup \{\varnothing\}) \times ([n] \cup \{\varnothing\})$. In this case, the SDP constraint reduces to

$$\langle u_i, u_j\rangle = \widetilde{P}_{ij}(X_i = X_j = 1)$$

for all $(i,j) \in ([n] \cup \{\varnothing\}) \times ([n] \cup \{\varnothing\})$.

We first investigate the objective. We see that

$$\begin{aligned} \|u_i - u_j\|_2^2 &= \|u_i\|_2^2 + \|u_j\|_2^2 - 2\langle u_i, u_j\rangle \\ &= \widetilde{P}_i(X_i = 1) + \widetilde{P}_i(X_j = 1) - 2\widetilde{P}_{ij}(X_i = X_j = 1) \\ &= \widetilde{P}_{ij}(X_i = 1) + \widetilde{P}_{ij}(X_j = 1) - 2\widetilde{P}_{ij}(X_i = X_j = 1) = \widetilde{P}_{ij}(X_i \neq X_j). \end{aligned}$$

Also, observe that

$$\langle u_i, u_j\rangle = \mathbb{E}_{\widetilde{P}_{ij}}[x_i x_j],$$

hence we create a matrix $M \in \mathbb{R}^{([n]\cup\{\varnothing\})\times([n]\times\{\varnothing\})}$ such that $M_{ij} = \mathbb{E}_{\widetilde{P}_{ij}}[x_i x_j]$, i.e., $M = UU^\top$. In this case, the original constraint implies that M is PSD, hence overall, the SDP becomes

$$\begin{aligned} & \max \sum_{(i,j)\in\mathcal{E}} \widetilde{P}_{ij}(X_i \neq X_j) \\ & \{\widetilde{P}_i\} \cup \{\widetilde{P}_{ij}\} \text{ is a 2-local distribution} \\ (\mathcal{P}) \quad & M = \left(\mathbb{E}_{\widetilde{P}_{ij}}[X_i X_j]\right)_{i,j\in[n]\cup\{\varnothing\}} \succeq 0, \end{aligned}$$

where we call this SDP $\mathcal{P}$.

We see that the notion of 2-local distribution can be generalized to arbitrary number R, i.e., we can now look at the so-called R-local distribution.

Definition 5.2.3 (Local distribution). The set of distributions $\{\widetilde{P}_A\}_{A\subseteq[n]\cup\{\varnothing\},|A|\leq R}$ is an *R-local distribution* if for all $A, B \subseteq [n] \cup \{\varnothing\}$ with $|A \cup B| \leq R$, for all $C \subseteq A \cup B$ and $\theta_i \in \{0,1\}$,

$$\widetilde{P}_C(X_i = \theta_i\ \forall i \in C) = \widetilde{P}_A(X_i = \theta_i\ \forall i \in C) = \widetilde{P}_B(X_i = \theta_i\ \forall i \in C).$$

Note. Notice that we can also define the generalized version of 2-local consistency, but we just encoded this in Definition 5.2.3.

Now, the R-local version of $\mathcal{P}$ becomes

$$\begin{aligned} & \max \sum_{(i,j)\in\mathcal{E}} \widetilde{P}_{ij}(X_i \neq X_j) \\ & \{\widetilde{P}_A\}_{|A|\leq R} \text{ is an } R\text{-local distribution} \\ \mathrm{Lass}_R(\mathcal{P}) \quad & M = \left(\mathbb{E}_{\widetilde{P}_{A\cup B}}\left[\prod_{i\in A\cup B} X_i\right]\right)_{A,B} \succeq 0, \end{aligned}$$

where we call this SDP $\mathrm{Lass}_R(\mathcal{P})$, which is how we define Lasserre hierarchy.

Note. Notice that $M \in \mathbb{R}^{\binom{[n]}{\leq R/2} \times \binom{[n]}{\leq R/2}}$

Lecture 16: Lasserre Hierarchy Continued

After defining $\mathrm{Lass}_R(\mathcal{P})$, we now analyze what kind of properties this hierarchy has. 26 Oct. 10:30

Remark. Up to this time, we have seen the following.

(a) $\mathrm{Lass}_R(\mathcal{P})$ is a convex program with $2^R n^{O(R)}$ variables and constraints.

(b) We can solve this in $n^{O(R)}$ time.

(c) $\mathrm{Lass}_2(\mathcal{P})$ is equivalent to a basic SDP, and $\mathrm{Lass}_n(\mathcal{P})$ is equivalent to an IP.

5.2.2 Probabilistic Consequences

Consider

- $\mathrm{Var}_{\widetilde{P}_i}[X_i] = \mathbb{E}_{\widetilde{P}_i}\left[X_i^2\right] - \mathbb{E}_{\widetilde{P}_i}[X_i]^2$
- $\mathrm{Cov}_{\widetilde{P}_{ij}}[X_i, X_j] = \mathbb{E}_{\widetilde{P}_{ij}}[X_i X_j] - \mathbb{E}_{\widetilde{P}_i}[X_i]\,\mathbb{E}_{\widetilde{P}_j}[X_j]$.

We see that we can do a conditioning: let $\widetilde{P} \coloneqq \{\widetilde{P}_A\}_{|A| \leq R}$ be an R-local distribution. Now, fix $S \subseteq [n]$, $|S| = t$, let $\alpha_S = \{0,1\}^{|S|}$, condition on $\widetilde{P}$, we get

$$\widetilde{P}' = \widetilde{P} \mid X_S \leftarrow \alpha_S,$$

where $X_S \leftarrow \alpha_S$ means $X_i \leftarrow \alpha_i$ for all $i \in S$.

Remark. $\widetilde{P}'$ is a $(R-t)$-local distribution.

Proof. For all $A \subseteq [n]$, $|A| \leq R - t$, we have

$$\widetilde{P}'_A(X_A = \theta_A) = \frac{\widetilde{P}(X_A = \theta_A, X_S = \alpha_S)}{\widetilde{P}(X_S = \alpha_S)}.$$

⊛

Apart from this, we also see that

(a) $\widetilde{P}'$ is $(R-t)$-wise locally consistent.

(b) If $\widetilde{P}$ was $\mathrm{Lass}_R(\mathcal{P})$, $\widetilde{P}'$ is feasible for $\mathrm{Lass}_{R-t}(\mathcal{P})$.

Lemma 5.2.1 (Conditioning reduces variance)**.** For all $i, j \in [n]$,

$$\mathrm{Var}_{\widetilde{P}_i}[X_i] - \mathbb{E}_{X_j \sim \widetilde{P}_j}\left[\mathrm{Var}_{\widetilde{P}_{ij}}[X_i \mid X_j]\right] \geq 4\,\mathrm{Cov}_{\widetilde{P}_{ij}}[X_i, X_j]^2.$$

Proof. From of law of total variance, we have

$$\mathrm{Var}_{\widetilde{P}_i}[X_i] - \mathbb{E}_{X_j}\left[\mathrm{Var}_{\widetilde{P}}[X_i \mid X_j]\right] = \mathrm{Var}_{\widetilde{P}_j}\left[\mathbb{E}_{\widetilde{P}_j}[X_i \mid X_j]\right].$$

Now, let $P_i = \widetilde{P}_i(X_i = 1)$, $P_j = \widetilde{P}_j(X_j = 1)$, $P_{ij} = \widetilde{P}_{ij}(X_i = 1, X_j = 1)$, we have

$$\begin{aligned}
\operatorname{Var}_{X_j}\left[\mathbb{E}_{\widetilde{P}}\left[X_i \mid X_j\right]\right] &= \mathbb{E}_{X_j}\left[\mathbb{E}_{\widetilde{P}}\left[X_i \mid X_j\right]^2\right] - \left(\mathbb{E}_{X_j}\left[\mathbb{E}\left[X_i \mid X_j\right]\right]\right)^2 \\
&= \widetilde{P}_j(X_j = 1) \cdot \frac{\mathbb{E}_{\widetilde{P}}\left[X_i X_j\right]^2}{\widetilde{P}_j(X_j = 1)^2} + \widetilde{P}_j(X_j = 0) \cdot \frac{\mathbb{E}_{\widetilde{P}}\left[X_i(1 - X_j)\right]^2}{\widetilde{P}_j(X_j = 0)^2} - \mathbb{E}_{\widetilde{P}}\left[X_i\right]^2 \\
&= \frac{P_{ij}^2}{P_j} + \frac{(P_i - P_{ij})^2}{1 - P_j} - P_i^2 \\
&= \frac{1}{P_j(1 - P_j)}\left(P_{ij}^2(1 - P_j) + (P_i - P_{ij})^2 \cdot P_j - P_i^2 P_j(1 - P_j)\right) \\
&= \frac{(P_{ij} - P_i P_j)^2}{P_j(1 - P_j)} \\
&= \frac{\left(\mathbb{E}_{\widetilde{P}}\left[X_i X_j\right] - \mathbb{E}_{\widetilde{P}_i}\left[X_i\right]\mathbb{E}_{\widetilde{P}_j}\left[X_j\right]\right)^2}{\mathbb{E}\left[X_j^2\right] - \mathbb{E}\left[X_j\right]^2} \\
&= \frac{\operatorname{Cov}_{\widetilde{P}}\left[X_i, X_j\right]^2}{\operatorname{Var}_{\widetilde{P}_i}\left[X_j\right]}.
\end{aligned}$$

Since X_i are 0-1 variable, the variance in the denominator is less than $1/4$, hence we finally have

$$\operatorname{Var}_{\widetilde{P}_i}\left[X_i\right] - \mathbb{E}_{X_j}\left[\operatorname{Var}_{\widetilde{P}}\left[X_i \mid X_j\right]\right] = \operatorname{Var}_{X_j}\left[\mathbb{E}_{\widetilde{P}}\left[X_i \mid X_j\right]\right] \geq 4 \cdot \operatorname{Cov}_{\widetilde{P}}\left[X_i, X_j\right]^2.$$

■

Corollary 5.2.1. Suppose $\widetilde{P} = \{\widetilde{P}_A\}_{|A| \leq R}$ is an R-local distribution which is $\mathrm{Lass}_R(\mathcal{P})$ feasible, then

$$\mathbb{E}_{j \sim [n]}\mathbb{E}_{X_j \sim \widetilde{P}_j}\left[\mathbb{E}_{i \sim [n]}\left[\operatorname{Var}_{\widetilde{P}_i}\left[X_i\right]\right] - \mathbb{E}_{i \sim [n]}\left[\operatorname{Var}_{\widetilde{P}_{ij}}\left[X_i \mid X_j\right]\right]\right] \geq 4\mathbb{E}_{i,j \sim [n]}\left[\operatorname{Cov}_{\widetilde{P}_{ij}}\left[X_i, X_j\right]^2\right].$$

Furthermore, given $a \in \mathbb{R}^+$, either one of the following will happen.

(a) $\mathbb{E}_{i,j \sim [n]}\left[\operatorname{Cov}_{\widetilde{P}_{ij}}\left[X_i, X_j\right]^2\right] \leq a$.

(b) $\exists j \in [n]$, $\theta_j \in \{0, 1\}$, $\widetilde{P}' := \widetilde{P} \mid X_j \leftarrow \theta_j$ satisfies $\mathbb{E}_{i \sim [n]}\left[\operatorname{Var}_{\widetilde{P}_i}\left[X_i\right]\right] - \mathbb{E}_{i \sim [n]}\left[\operatorname{Var}_{\widetilde{P}'}\left[X_i\right]\right] \geq 4a$.

Proof. We first prove the first statement. Lemma 5.2.1 gives a point-wise inequality, taking the expectation on both sides with the dominanted convergence theorem, we have

$$\mathbb{E}_{i,j \sim [n]}\left[\operatorname{Var}\left[X_i\right] - \mathbb{E}_{X_j}\left[\operatorname{Var}\left[X_i \mid X_j\right]\right]\right] \geq 4\mathbb{E}_{i,j \sim [n]}\left[\operatorname{Cov}\left[X_i, X_j\right]^2\right],$$

with the fact that

$$\mathbb{E}_{i,j \sim [n]}\left[\operatorname{Var}\left[X_i\right] - \mathbb{E}_{X_j}\left[\operatorname{Var}\left[X_i \mid X_j\right]\right]\right] = \mathbb{E}_{j \sim [n]}\mathbb{E}_{X_j}\left[\mathbb{E}_{i \sim [n]}\left[\operatorname{Var}\left[X_i\right]\right] - \mathbb{E}_{i \sim [n]}\left[\operatorname{Var}\left[X_i \mid X_j\right]\right]\right],$$

hence conclude the first part. A probabilistic argument proves the *either-or* statement. ■

Remark. Corollary 5.2.1 says that either we have a small covariance, or we can reduce it by a lot.

Theorem 5.2.1. Suppose $\widetilde{P} = \{\widetilde{P}_A\}_{|A| \leq R}$ is an R-local distribution which is $\mathrm{Lass}_R(\mathcal{P})$ feasible and $R \geq 1/\epsilon^4 + 2$, then there exists $S \subseteq [n]$ such that $|S| \leq 1/\epsilon^4$, $\alpha_S \in \{0, 1\}^{|S|}$, and $\widetilde{P}' := \widetilde{P} \mid X_S \leftarrow \alpha_S$, we have

$$\mathbb{E}_{ij \sim [n]}\left[\operatorname{Cov}_{\widetilde{P}'}\left[X_i, X_j\right]^2\right] \leq \frac{\epsilon^4}{4}.$$

Moreover, S and α_S can be found in $\mathsf{poly}(n, 1/\epsilon)$.

Proof. We actually have a constructive proof, i.e., we directly give an algorithm which runs in $\mathsf{poly}(n, 1/\epsilon)$ and find the desired i, j.

Algorithm 5.2: Theorem 5.2.1 – Construction

Data: $\widetilde{P}$, $\epsilon > 0$
Result: $\widetilde{P}'$ with expected covariance smaller than $\epsilon^4/4$, S

1 $\ell \leftarrow 0$, $\widetilde{P}^{(\ell)} \leftarrow \widetilde{P}$, $S \leftarrow \varnothing$
2 **for** $\ell = 0, 1, \ldots, 1/\epsilon^4$ **do**
3   **if** $\mathbb{E}_{i,j}\left[\mathrm{Cov}_{\widetilde{P}^{(\ell)}}[X_i, X_j]^2\right] \leq \epsilon^4/4$ **then**
4   **return** $\widetilde{P}^{(\ell)}$ // $\widetilde{P}^{(\ell)} = \widetilde{P} \mid X_S \leftarrow \alpha_S$, S
5   **else**
6   Find $j_{\ell+1} \in [n] \setminus S$, $\theta_{\ell+1} \in \{0, 1\}$ // Guaranteed in Corollary 5.2.1
7     $\widetilde{P}^{(\ell+1)} \leftarrow \widetilde{P}^{(\ell)} \mid X_{j_{\ell+1}} \leftarrow \theta_{\ell+1}$
8     $S \leftarrow S \cup \{j_{\ell+1}\}$

To analyze Algorithm 5.2, observe that if Algorithm 5.2 returns, then we have a desired property, so we only need to ensure it'll meet the condition in line 3 in $1/\epsilon^4$ iterations. Now, for a local distribution Q, let $\mathrm{Var}[Q] \coloneqq \mathbb{E}_{i\sim[n]}[\mathrm{Var}_Q[X_i]]$ and $\mathrm{Cov}[Q] \coloneqq \mathbb{E}_{i,j\sim[n]}[\mathrm{Cov}_Q[X_i, X_j]]$. We see that we only fail if in every iteration, we reach line 5, i.e., $\mathrm{Cov}[\widetilde{P}^{(\ell)}] \leq \epsilon^4/4$ for all ℓ. But from Corollary 5.2.1, we know that the $\widetilde{P}^{(\ell+1)}$ we find will have the property that

$$\mathrm{Var}[\widetilde{P}^{(\ell-1)}] - \mathrm{Var}[\widetilde{P}^{(\ell)}] \geq 4 \cdot \epsilon^4/4 = \epsilon^4 \Rightarrow \mathrm{Var}[\widetilde{P}^{(\ell)}] \leq \mathrm{Var}[\widetilde{P}^{(\ell-1)}] - \epsilon^4.$$

By telescoping, we have

$$\mathrm{Var}[\widetilde{P}^{(1/\epsilon^4)}] \leq \mathrm{Var}[\widetilde{P}^{(0)}] - \frac{1}{\epsilon^4} \cdot \epsilon^4 = \mathrm{Var}[\widetilde{P}] - 1 \leq \frac{1}{4} - 1 < 0,$$

a contradiction, and hence we must terminate, finishing the proof. ∎

Remark. Theorem 5.2.1 says that suppose we have a local distribution over n variables with sufficient large locality. Then turns out that there's a small subset of variables, if we fix them, they'll almost determine all other variables.

Finally, we have the following algorithm.

Algorithm 5.3: Max Cut – PTAS

Data: A dense graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{E}| \geq \epsilon n^2$, $\epsilon > 0$
Result: A cut S

1 $R \leftarrow 1/\epsilon^4 + 2$
2 $\widetilde{P} \coloneqq \{\widetilde{P}_A\}_{|A| \leq R} \leftarrow$`Solve`$(\mathrm{Lass}_R(\mathcal{P}))$
3 $\widetilde{P}' \leftarrow$`Reduce-Variance`$(\widetilde{P})$ // $\mathbb{E}_{ij\sim[n]}\left[\mathrm{Cov}_{\widetilde{P}'}[X_i, X_j]^2\right] \leq \epsilon^4/4$
4
5 `// Rounding`
6 **for** $i \in \mathcal{V}$ **do**
7   $\lambda_i \leftarrow$`Ber`$(\widetilde{P}'(X_i = 1))$
8 $S \leftarrow \{i \in \mathcal{V} \colon \lambda_i = 1\}$
9 **return** S

Remark. The rounding method in Algorithm 5.3 (i.e., line 4) is ridiculously simple compare to Algorithm 5.1! $\widetilde{P}'$ basically tells you everything.

We indeed have the following guarantee.

Theorem 5.2.2 (PTAS for max cut)**.** For any $\epsilon > 0$, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that $|\mathcal{E}| \geq \epsilon n^2$, there exists a $(1 - 4\epsilon)$-approximation algorithm runs in $n^{O(1/\epsilon^4)}$-time.

We see that as long as the graph is dense enough, we can spend more and more time to get a better approximation, which is the whole point of Lasserre hierarchy.

Lecture 17: Graph Coloring

Let's first prove Theorem 5.2.2. 31 Oct. 10:30

Proof of Theorem 5.2.2. The running time for Algorithm 5.3 is clear. Denote $p_i = \widetilde{P}'(X_i = 1)$, then we see that the expected fraction of edges cuts is

$$
\begin{aligned}
&\mathbb{E}_{(i,j)\in\mathcal{E}}\left[\Pr_{\mathrm{ALG}}(X_i \neq X_j)\right] \\
=&\mathbb{E}_{X_S}\mathbb{E}_{(i,j)\in\mathcal{E}}\left[p_i + p_j - 2p_ip_j\right] \\
\geq&\mathbb{E}_{X_S}\mathbb{E}_{(i,j)\in\mathcal{E}}\left[p_i + p_j - 2\mathbb{E}_{\widetilde{P}'}\left[X_iX_j\right] - 2\left|p_ip_j - \mathbb{E}_{\widetilde{P}'}\left[X_iX_j\right]\right|\right] \\
=&\underbrace{\mathbb{E}_{(i,j)\in\mathcal{E}}\left[\mathbb{E}_{\widetilde{P}}\left[X_i\right] + \mathbb{E}_{\widetilde{P}}\left[X_j\right] - 2\mathbb{E}_{\widetilde{P}}\left[X_iX_j\right]\right]}_{\text{Lass}} - \underbrace{2\mathbb{E}_{X_S}\mathbb{E}_{(i,j)\in\mathcal{E}}\left[p_ip_j - \mathbb{E}_{\widetilde{P}'}\left[X_iX_j\right]\right]}_{\text{Err}}.
\end{aligned}
$$

Recall that previously, we only have control on $\mathrm{Cov}^2_{\widetilde{P}'} = \mathrm{Cov}^2(\widetilde{P} \mid X_S \leftarrow \alpha_S)$, which is over the whole $i, j \sim [n]$. But now the error term (the second term) is only over $(i,j) \sim \mathcal{E}$, hence we define

$$
\mathrm{Cov}^2_{\mathcal{E}}\left[\widetilde{P} \mid X_S \leftarrow \alpha_S\right] = \mathbb{E}_{X_S}\mathbb{E}_{(i,j)\sim\mathcal{E}}\left[\mathrm{Cov}\left[X_i, X_j \mid S_S\right]^2\right].
$$

Claim. $\mathrm{Cov}^2_{\mathcal{E}}[\widetilde{P} \mid S] \leq \mathrm{Cov}[\widetilde{P} \mid X_S \leftarrow \alpha_S]/\epsilon \leq \epsilon^3$.

Proof. We see that

$$
\begin{aligned}
n^2 \cdot \mathrm{Cov}^2\left[\widetilde{P} \mid X_S \leftarrow \alpha_S\right] &= \mathrm{Cov}^2_{\Sigma}\left[\widetilde{P} \mid X_S \leftarrow \alpha_S\right] \\
&\geq \mathrm{Cov}^2_{\mathcal{E},\Sigma}\left[\widetilde{P} \mid X_S \leftarrow \alpha_S\right] = m \cdot \mathrm{Cov}^2_{\mathcal{E}}\left[\widetilde{P} \mid X_S \leftarrow \alpha_S\right],
\end{aligned}
$$

where subscript Σ is when we replace the expectation by summation in the covariance. With the fact that $m \leftarrow \epsilon n^2$, we're done. ⊛

Now, since we know that Lass $\geq$ OPT $\geq 1/2$, we have

$$
\mathbb{E}_{(i,j)\in\mathcal{E}}\left[\Pr_{\mathrm{ALG}}(X_i \neq X_j)\right] \geq \text{Lass} - 2\epsilon \geq \text{Lass}(1 - 4\epsilon),
$$

finishing the proof. ■

5.3 Graph Coloring

Return to the SDP, first, we introduce a new definition.

Definition 5.3.1 (Coloring)**.** Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a *(valid) coloring* $\chi\colon \mathcal{V} \to [c]$ is a function χ such that for all $(i,j) \in \mathcal{E}$, $\chi(i) \neq \chi(j)$.

Now, consider the following problem.

Problem 5.3.1 (Graph coloring)**.** Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, find a coloring $\chi\colon \mathcal{V} \to [c]$ while minimizing c.

Before trying to solve the graph coloring, we note that it's trivial to get n-coloring (by using different color for every node). But in fact, this is the best we can do: graph coloring is extremely hard!

Theorem 5.3.1. For all $\epsilon > 0$, it's NP to get $n^{1-\epsilon}$-approximation.

People start to consider some promise version of graph coloring, i.e., if we directly assume that $\mathcal{G}$ admits a c-coloring, what can we say?

- $c = 1$: $\mathcal{G}$ has no edges, hence trivial.
- $c = 2$: $\mathcal{G}$ is bipartite, so we can color the graph alternatively, so this can be solved exactly.
- $c = 3$: We can do $\widetilde{O}(n^{1/4})$-approximation (quite shameful...)

Remark (SOTA for $c = 3$)**.** For $c = 3$, someone showed that we can do $\widetilde{O}(n^{0.199\cdots})$, i.e., around $\widetilde{O}^{1/5}$. Also, $\omega(1)$-approximation is NP!

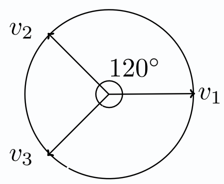
Analogous to max cut, we design the following SDP relaxation of graph coloring with variables being vectors v_i for $i \in \mathcal{V}$.

$$
\begin{aligned}
\min\ & 0 \\
& \langle v_i, v_i \rangle = 1 && \forall i \in \mathcal{V} \\
& \langle v_i, v_j \rangle = -1/2 && \forall (i, j) \in \mathcal{E}.
\end{aligned}
\tag{5.2}
$$

Note. We don't have an actual objective function!

Claim. If $\mathcal{G}$ is 3-colorable, there exists $\{v_i\}_{i \in \mathcal{V}}$ that are feasible for Equation 5.2.

Proof. Consider 3 colors C_1, C_2, and C_3, then if i has C_j, we let i gets vector v_j with all vectors v_j, $v_{j'}$ are $120°$ away.



⊛

5.3.1 Independent Sets

Turns out that the independent set is highly related to solving just Equation 5.2, as we'll soon see.

Definition 5.3.2 (Independent set)**.** Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a set $S \subseteq \mathcal{V}$ is *independent* if for all $(i, j) \in \mathcal{E}$, either $i \notin S$ or $j \notin S$.

The notion of independent set is useful since we can transform the graph coloring problem into finding independent sets as suggests by Lemma 5.3.1.

Lemma 5.3.1. For a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, then $\mathcal{G}$ is c-colorable if and only if $\mathcal{V}$ can be partitioned to $V_1, \ldots, V_c$ such that V_i is independent.

With Lemma 5.3.1, if we can find large independent sets and partition the graph into not too many of those, we're done. Note that the size of independent sets is related to the maximum degree.

Notation. We denote the maximum degree of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by $\Delta \coloneqq \Delta(\mathcal{G}) \coloneqq \max_{v \in \mathcal{V}} \deg(v)$.

Remark (Usefullness of Δ). Given Δ, we will have a trivial Δ-coloring; also, we know that we can find independent set with size $n/(\Delta + 1) = \Omega(n\Delta^{-1})$.

Proof. The coloring part is clear. As for finding independent set, we see that by randomly include one vertex to our independent set, we at most Δ vertices will be ruled out: they can't be in the independent set now. ⊛

Now, after solving Equation 5.2, notice that we only have feasibility, with the fact that the solution are not guaranteed to be perfectly aligned in exactly three vectors, hence it's a bit confusing what to do next. However, recall that it's also good enough to find a large independent set, and recall the max cut problem, where we want to maximize the number of edges crossing a cut set, which is similar to what we're trying to do here. So, inspired by which, we can round the solution, but observe the following.

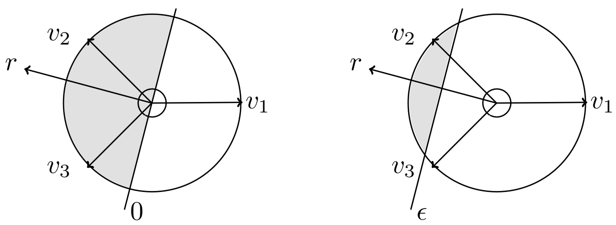


Figure 5.1: If we do the rounding as in max cut, we may end-up including more than we want, so we set up some threshold.

This suggests that we round it with threshold, i.e., consider the following algorithm.

Algorithm 5.4: Graph Coloring – Independent Set Rounding of 3-Colorable Graph

Data: A 3-colorable graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Result: An independent set S

1 $\{v_i \in \mathbb{R}^d\}_{i=1}^n \leftarrow$ `Solve(`*SDP*`)`
2 $r \leftarrow \mathcal{N}(0, I_d)$ // $r_i \sim \mathcal{N}(0,1)$
3 $S(\epsilon) \leftarrow \{i \in \mathcal{V} \colon \langle r, v_i \rangle \geq \epsilon\}$ // $\epsilon = \sqrt{2/3 \cdot \ln \Delta}$
4 $S'(\epsilon) = \{i \in S(\epsilon) \colon \nexists j \in S(\epsilon) \text{ s.t. } (i,j) \in \mathcal{E}\}$ // Make S independent
5 **return** $S'(\epsilon)$

Remark. Algorithm 5.4 is the rounding algorithm of Equation 5.2 in the sense of feasibility,[a] and notice that it gives us an independent set, rather than a coloring.

[a] Recall that there's no objective in Equation 5.2

Lecture 18: Graph Coloring via Independent Sets Decomposition

We're now interested in how large the independent set Algorithm 5.4 outputs. To start analyzing, since 2 Nov. 10:30
the r sampled in line 2 is Gaussian, recall the following.

As previously seen (Gaussian distribution). The probability density function for Gaussian distribution is

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2},$$

and the cumulated density function is

$$\Phi(x) = \Pr_g(g \leq x) = \int_{-\infty}^{x} p(x)\,\mathrm{d}s, \quad \overline{\Phi}(x) = \Pr_g(g \geq x) = 1 - \Phi(x).$$

From the spherical symmetry, we have $\langle r, v_i \rangle \sim \mathcal{N}(0,1)$ for all $i \in \mathcal{V}$. Moreover, since the probability of i being in $S(\epsilon)$ is exactly $\overline{\Phi}(\epsilon)$, from the linearity of expectation, we have $\mathbb{E}\left[|S(\epsilon)|\right] = n \cdot \overline{\Phi}(\epsilon)$.

Lemma 5.3.2. $\Pr(i \notin S'(\epsilon) \mid i \in S(\epsilon)) \leq \Delta\overline{\Phi}(\sqrt{3}\epsilon)$.

Proof. Fix any $(i,j) \in \mathcal{E}$, it's sufficient to show $\Pr(j \in S(\epsilon) \mid i \in S(\epsilon)) \leq \overline{\Phi}(\sqrt{3}\epsilon)$. And from the fact that all v_j are 120° apart, we hence can write

$$v_j = -\frac{1}{2}v_i + \frac{\sqrt{3}}{2}u$$

where $\|u\| = 1$ and $u \perp v_i$. If $j \in S(\epsilon)$, then

$$\langle v_j, r \rangle \geq \epsilon \Rightarrow \underbrace{\left\langle -\frac{1}{2}v_i, r \right\rangle}_{\leq -\epsilon/2} + \left\langle \frac{\sqrt{3}}{2}u, r \right\rangle \geq \epsilon \Rightarrow \left\langle \frac{\sqrt{3}}{2}u, r \right\rangle \geq \frac{3\epsilon}{2} \Rightarrow \langle u, r \rangle \geq \sqrt{3}\epsilon.$$

Since if $u \perp v \in \mathbb{R}^d$, $\langle u, r \rangle$ and $\langle v, r \rangle$ are independent, so $\Pr(j \in S(\epsilon) \mid i \in S(\epsilon)) \leq \overline{\Phi}(\sqrt{3}\epsilon)$ as desired. ■

We can now prove that the independent set found by Algorithm 5.4 is large.

Theorem 5.3.2. Algorithm 5.4 finds an independent set of size $\Omega(n \cdot \Delta^{-1/3} \log^{-1/2} \Delta)$ for any 3-colorable $\mathcal{G}$.

Proof. We see that

$$\mathbb{E}\left[|S'(\epsilon)|\right] = \sum_{i \in \mathcal{V}} \underbrace{\Pr(i \in S(\epsilon))}_{\overline{\Phi}(\epsilon)} \cdot \Pr(i \in S'(\epsilon) \mid i \in S(\epsilon)) \geq \sum_{i \in \mathcal{V}} \overline{\Phi}(\epsilon) \cdot \left(1 - \Delta\overline{\Phi}(\sqrt{3}\epsilon)\right),$$

from Lemma 5.3.2. Now, observe the following.

Claim. If $x \geq 10$, $p(x)/2x \leq \overline{\Phi}(x) \leq p(x)/x$.

Proof. Since we know that

$$\frac{x}{1+x^2} \cdot p(x) \leq \overline{\Phi}(x) \leq \frac{1}{x} \cdot p(x)$$

for all x, if $x \geq 10$, $x/(1+x^2) \geq 1/2x$, hence we're done. ⊛

With the above claim, we have $\overline{\Phi}(\epsilon) \geq p(\epsilon)/2\epsilon$ with $\overline{\Phi}(\sqrt{3}\epsilon) \leq p(\sqrt{3}\epsilon)/3\epsilon$, hence

$$p(\sqrt{3}\epsilon) = \frac{1}{\sqrt{2\pi}} e^{-3\cdot(2/3\cdot\ln\Delta)/2} = \frac{1}{\sqrt{2\pi}} \cdot \frac{1}{\Delta} \text{ and } p(\epsilon) = \frac{1}{\sqrt{2\pi}} e^{-2/3\cdot\ln\Delta/2} = \frac{1}{\sqrt{2\pi}} \frac{1}{\Delta^{1/3}},$$

leading to

$$\mathbb{E}\left[|S'(\epsilon)|\right] \geq \sum_{i \in \mathcal{V}} \underbrace{\left(\frac{1}{2\epsilon} \cdot p(\epsilon)\right)}_{\geq \Omega\left(\frac{1}{\sqrt{\ln\Delta}} \frac{1}{\Delta^{1/3}}\right)} \cdot \underbrace{\left(1 - \Delta \cdot \frac{1}{3\epsilon} \cdot p(\sqrt{3}\epsilon)\right)}_{\geq 1/2} \geq \Omega(n \cdot \Delta^{-1/3} \ln^{-1/2} \Delta).$$

■

5.3.2 Independent Sets Decomposition

From Lemma 5.3.1, we can iteratively find large independent sets by Algorithm 5.4 as guaranteed by Theorem 5.3.2. But before we see the final algorithm, we introduce a cute trick.

Remark (Wigderson's trick). We can always 3-color $\{v\} \cup \mathcal{N}(v)$ for all $v \in \mathcal{V}$ if $\mathcal{G}$ is 3-colorable.

Proof. Since for a 3-colorable graph, for all $v \in \mathcal{V}$, $\mathcal{G}[N(v)]$ is bipartite (all $u \in N(v)$ already links with v, so the degree will be at most 2 in $\mathcal{G}[\mathcal{N}(v)]$). And as mentioned before, we can always 2-colors a bipartite graph. And we just use another new color for v to do the 3-coloring. ⊛

Now, we see the final algorithm.

Algorithm 5.5: Graph Coloring – Independent Set Decomposition of 3-Colorable Graph

Data: A 3-colorable graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Result: A colored $\mathcal{G}$

$n_0 \leftarrow |\mathcal{V}|$

`// Phase 1`
while $\Delta(\mathcal{G}) \geq n_0^{3/4}$ **do**
  $v \leftarrow \arg\max_{i \in \mathcal{V}} \deg(i)$
  3 colors $\{v\} \cup N(v)$ `// Wigderson's trick`
  $\mathcal{G} \leftarrow \mathcal{G}[\mathcal{V} \setminus (\{v\} \cup N(v))]$

`// Phase 2`
while $\Delta(\mathcal{G}) \geq 100$ **do** `//` $\Delta(\mathcal{G}) < n_0^{3/4}$
  $S \leftarrow$`Independent-Set(`$\mathcal{G}$`)` `//` $|S| \geq cn\Delta^{-1/3} \ln^{-1/2} \Delta$ `from Theorem 5.3.2`
  1 colors S
  $\mathcal{G} \leftarrow \mathcal{G}[\mathcal{V} \setminus S]$

`// Phase 3`
$\Delta(\mathcal{G}) + 1$ colors $\mathcal{G}$ `//` $\Delta(\mathcal{G}) < 100$
return $\mathcal{G}$

Notice that in Algorithm 5.5, whenever we do a coloring, we use a brand-new color to avoid any collision.

Theorem 5.3.3. Algorithm 5.5 is an $\widetilde{O}(n^{1/4})$-approximation algorithm for graph coloring.

Proof. We see that

- Phase 1: color at least $n_0^{3/4}$ vertices with 3 colors in each iteration, hence need at most $n_0^{1/4} (= 3 \cdot n_0 / n_0^{3/4})$ colors.
- Phase 2: from Theorem 5.3.2, $|S| \geq ncn_0^{-1/4} \log^{-1/2} n_0 =: n\gamma$, then the induced graph will have vertices less than $n(1-\gamma)$.[a] Hence, we can run this at most k iteration since $n \geq 1$, i.e.,

$$1 \leq n \leq n_0(1-\gamma)^k \leq n_0 e^{-\gamma k} \Rightarrow k \leq \frac{1}{\gamma} \ln n_0 = \frac{1}{c} n_0^{1/4} \log^{1/2} n_0 \cdot \ln n_0$$

- Phase 3: Clean-up phase, only uses constant amount (< 100) more colors.

In all, Algorithm 5.5 uses at most

$$3n_0^{1/4} + \frac{1}{c} n_0^{1/4} \log^{3/2} n_0 + 100 = \widetilde{O}(n^{1/4})$$

colors. ■

[a] Notice the different between n and n_0: n is updating, while n_0 is the original graph size.

Remark. We can use the similar approach for small constant c, e.g., $c = 4$, $c = 5$, etc.

Chapter 6

Hardness of Approximation

Lecture 19: Complexity Theory for Approximation Algorithm

Recall how we define the combinatorial optimization. 7 Nov. 10:30

> **As previously seen.** Given a set of all possible inputs $\mathcal{I}$ for a combinatorial optimization problem P, the goal is to find $x \in X_I$ to maximize/minimize $f_I(x)$ where $f_I \colon X_I \to \mathbb{R}^+$ is an objective function, and X_I is a set of feasible solutions.

Now, we're going to discuss the complexity of doing approximation problem. But since the classical complexity theory is under the context of decision problems, we now try to generalize it.

6.1 Approximation Complexity

In this section, we'll consider maximization problems primarily, however, the same definition can be adapted to minimization problems naturally. First consider the decision problems: given a maximization problem P with goal being finding an objective in $\mathbb{R}$, we have the following decision version.

Definition 6.1.1 (Decision-P). Given a maximization problem P, the *decision-P* is the decision version of P, where given an input $I \in \mathcal{I}$, $c \in \mathbb{R}^+$, finds an algorithm which output True if $\mathsf{OPT}_I \geq c$, False otherwise.

And we have the following characterization of P and decision-P in terms of complexity class.

Definition 6.1.2. A maximization problem P is NP if decision-P is NP.

6.1.1 The Gap Problem

Adapting this to the approximation version, we have the following generalization.

Definition 6.1.3 (Gap). Given a maximization problem P with $\alpha \leq 1$, the *α-Gap P* is the decision version of α-approximating P, where given an input $I \in \mathcal{I}$ and $c \in \mathbb{R}^+$, finds an algorithm which outputs True if $\mathsf{OPT}_I \geq c$, False if $\mathsf{OPT}_I < \alpha c$, and anything else (don't care) otherwise.

Intuition. Since we allow some approximation, we don't care about the "gap", i.e., $\mathsf{OPT}_I \in [\alpha c, c)$.

Since we see that we may ignore some outputs, we divide the output into two different sets.

Notation. Let $\mathcal{I}$ be a set of all inputs, Y be a set of all True inputs, N be a set of all False inputs.[a] Then given an input $I \in \mathcal{I}$, output True if $I \in Y$, False if $I \in N$, anything otherwise.

[a] We have $Y \cap N = \varnothing$, while not necessarily have $N \cup Y = \mathcal{I}$.

Remark. If there is an α-approximation algorithm for P, then there is an algorithm for α-Gap P.

Proof. Given I and c, we run the α-approximation algorithm for P to get a solution with value c'. Notice that we necessarily have

$$\alpha\,\mathsf{OPT}_I \leq c' \leq \mathsf{OPT}_I,$$

hence we can design a new algorithm which outputs True if $c' \geq c \cdot \alpha$, False otherwise. This is a correct algorithm for α-Gap P since

- If $\mathsf{OPT}_I \geq c$, then $c' \geq \alpha\,\mathsf{OPT}_I \geq \alpha c$, which is the True case.
- If $\mathsf{OPT}_I < \alpha c$, then $c' \leq \mathsf{OPT}_I < \alpha c$, which is the False case.

Conversely, it there is no polynomial time algorithm for α-Gap P, there is no α-approximation algorithm for P. ⊛

Again, we have the following characterization of P and α-Gap P in terms of complexity class.

Definition 6.1.4. An α-approximating problem P is NP if the α-Gap P is NP.

6.1.2 Approximation Reduction

Finally, we briefly review the classical reduction for decision problem.

As previously seen (Reduction). Given two problems P_1, P_2, a *reduction* from P_1 to P_2 is a polynomial time algorithm R such that given an input $I_1 \in \mathcal{L}_1$, output $I_2 \in \mathcal{L}_2$ satisfying both completeness and soundness.

> **As previously seen** (Completeness). A reduction from P_1 to P_2 satisfies *completeness* if it transforms an accepted input for P_1 to an accepted input for P_2, i.e., if $I_1 \in Y_1$, then $I_2 \in Y_2$.

> **As previously seen** (Soundness). A reduction from P_1 to P_2 satisfies *soundness* if it transforms a rejected input for P_1 to a rejected input for P_2, i.e., if $I_1 \in N_1$, then $I_2 \in N_2$.

The reason why we care about reduction is that given a reduction R from P_1 to P_2, if there exists an algorithm for P_2, then we have an algorithm for P_1. by the following.

Algorithm 6.1: Reduction

Data: Algorithm $\mathcal{A}_2$ for P_2, reduction R from P_1 to P_2, input $I_1 \in \mathcal{L}_1$ for P_1
Result: Decision of I_1

```
I_2 ← R(I_2)
return A_2(I_2)
```

Similarly, since we care about approximation algorithm, we can define the approximation version of the reduction from α-Gap P_1 to β-Gap P_2 as follows.

Definition 6.1.5 (Reduction). Given two maximization problems P_1, P_2, a *reduction* from α-Gap P_1 to β-Gap P_2 is a polynomial time algorithm R such that given an input $I_1 \in \mathcal{L}_1$ and c_1, output $I_2 \in \mathcal{L}_2$ and c_2 satisfying both completeness and soundness.

> **Definition 6.1.6** (Completeness). A reduction from P_1 to P_2 satisfies *completeness* if it transforms an accepted input for P_1 to an accepted input for P_2, i.e., if $\mathsf{OPT}_{I_1} \geq c_1$, then $\mathsf{OPT}_{I_2} \geq c_2$.

> **Definition 6.1.7** (Soundness). A reduction from P_1 to P_2 satisfies *soundness* if it transforms a rejected input for P_1 to a rejected input for P_2, i.e., if $\mathsf{OPT}_{I_1} < \alpha c_1$, then $\mathsf{OPT}_{I_2} < \beta c_2$.

Remark. The term completeness and soundness comes from logic.

Proof. More intuitively, for a proof system, completeness states that every true statement has a proof, while soundness states that every false statement can't have a proof, i.e., we can't prove anything that is wrong. ⊛

And again, given a reduction R if there is a polynomial time algorithm for β-Gap P_2, then we have a polynomial time algorithm for α-Gap P_1; on the other hand, if there is no polynomial time algorithm for α-Gap P_1, then there is no polynomial time algorithm for β-Gap P_2, so there is no β-approximation algorithm for P_2.

6.2 Probabilistically Checkable Proofs

6.2.1 Constraint Satisfaction Problem

We first study one of the most important problems in theoretical computer science, the CSP problem. This is important since it's the reduction for many important problems, and form the discussion, if we have a good algorithm for CSP, we automatically get lots of other problems solved.

Problem 6.2.1 (CSP). Given an input $(x_1, \ldots, x_n) = X$, $C_1, \ldots, C_m$ where $C_i = (a_i, b_{i_1}, \ldots, b_{i_k})$ be the set of clauses where $a_i \in \ell$, $b_{i_j} \in [n]$, the *constraint satisfaction problem of* Σ, Φ[a] is to find $\sigma\colon X \to \Sigma$ maximizing the number of satisfied clauses, i.e., $\sigma_{a_i}(x_{b_1}, \ldots, x_{b_k}) = 1$.

[a] Σ is the alphabet set and $\Phi = \{\phi_1, \ldots, \phi_\ell\}$ is a family of constraints where $\phi_i\colon \Sigma^k \to \{0,1\}$.

There's an important distinction between problem description and problem instance. That is, the CSP with respect to Σ, Φ is the problem description of a class of problems, and after given some variables X and clauses C_i, it becomes a problem instance, which can be solved.

Notation (Problem description). The problem description of CSP with respect to Σ and Φ is denoted as $\mathrm{CSP}(\Sigma, \Phi)$.

Notice that we can equivalently maximize the fraction instead of maximize the number of satisfied clauses, i.e., the objective is now $\#\text{satisfied clauses}/m$. It's because it's convenient to normalize the objective to be in $[0,1]$.

Note. Notice that to represent $\phi_i\colon \Sigma^k \to \{0,1\}$, it's often more convenient just to denote it as $\phi_i^{-1}(\{1\})$, i.e., the set of accepted string in Σ^k w.r.t. ϕ_i.

Example (Max-cut as CSP). Max cut is equivalent to $\mathrm{CSP}(\Sigma, \Phi)$ where $\Sigma = \{0,1\}$, $\Phi = \{\phi_1\}$ with $\phi_1 = \{01, 10\}$.

Proof. If we model max cut in this way, given an instance of max cut, i.e., given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n nodes, $C_i = (1, u, v)$ for $(u, v) \in \mathcal{E}$. The first entry is 1 since there are only one constraint to check whether a node is in the cut or not, and we create C_i for every edge (u, v). ⊛

Example (Max-2SAT as CSP). MAX-2SAT is equivalent to $\mathrm{CSP}(\Sigma, \Phi)$ where $\Sigma = \{0,1\}$, $\Phi = \{\phi_1, \ldots, \phi_4\}$ with

$$\phi_1 = \{01, 10, 11\} \Leftrightarrow (x_i \lor x_j), \quad \phi_2 = \{01, 10, 00\} \Leftrightarrow (\overline{x}_i \lor \overline{x}_j),$$
$$\phi_3 = \{01, 00, 11\} \Leftrightarrow (\overline{x}_i \lor x_j), \quad \phi_4 = \{00, 10, 11\} \Leftrightarrow (x_i \lor \overline{x}_j).$$

6.2.2 The Probabilistic Checkable Proofs Theorem

As mentioned, there's lots of reduction can be done between fundamental problems considered in TCS to CSP, including one of the most important results in hardness, the PCP theorem. In order to do this, we need a more fine-grained version of Definition 6.1.3.

Definition 6.2.1 ((c, s)-Gap)**.** Given a maximization problem P with $0 < s \leq c \leq 1$, the (c, s)-*Gap* P is the decision version of α-approximating P, where given an input $I \in \mathcal{I}$ and $c \in \mathbb{R}^+$, finds an algorithm which outputs True if $\mathsf{OPT}_I \geq c$, False if $\mathsf{OPT}_I < s$, and anything else (don't care) otherwise.

Note. We implicitly assume that (c, s)-Gap P is only defined for P being a CSP, or can be reduced to CSP.

Remark. We see that by setting $s = \alpha \cdot c$, we recover Definition 6.1.3 from Definition 6.2.1.

Then, we have the following.

Theorem 6.2.1 (Cook-Levin theorem [Coo71])**.** The $(1, 1)$-Gap 3SAT is NP-hard.

Theorem 6.2.2 (Karp [Kar72])**.** For all fixed $\epsilon > 0$, $(1 - \epsilon, 1 - \epsilon)$-Gap max cut is NP-hard.

Note. The $(1, 1)$-Gap max cut is P.

Proof. Recall that if we transform max cut into CSP, the optimal value is always 1,[a] i.e., every edge is cut edge, so in this case the graph must be bipartite. This can be easily check. ⊛

[a] i.e., we're not comparing to the optimal value of one instance of $\mathcal{G}$.

Theorem 6.2.3 (PCP theorem [Fei+91; Aro+98])**.** There exists an $\epsilon > 0$ such that $(1, 1 - \epsilon)$-Gap 3SAT is NP-hard.

To understand , we need to understand the class PCP. First, recall the definition of NP.

As previously seen (NP)**.** A language $L \subseteq \{0, 1\}^*$ is in NP if there exists a Turing machine V runs in $\mathsf{poly}(|x|)$ such that given x,

- $x \in L$, then $\exists y$ such that $V(x, y) = 1$;
- $x \notin L$, then $\forall y$ such that $V(x, y) = 0$.

Definition 6.2.2 (PCP)**.** The class *probabilistically checkable proofs*, or $\mathsf{PCP}_{c,s}(r(n), q(n))$,[a] is defined as $L \in \mathsf{PCP}_{c,s}(r, q)$ if there exists a poly-time randomized Turing machine V which can only flip r coins[b] and given an input x, V can look at x on q position $Q_1, \ldots, Q_q$ by $\phi_R\colon \{0, 1\}^q \to [0, 1]$ where

- $x \in L$, then $\exists y$ such that $\Pr_R(\phi(y_{Q_1}, \ldots, y_{Q_q}) = 1) \geq c$;
- $x \notin L$, then $\forall y$ such that $\Pr_R(\phi(y_{Q_1}, \ldots, y_{Q_q}) = 1) < s$.

[a] We implicitly assume that r and q depends on the length of the input $|x| = n$.
[b] It only accepts random string R with length r, i.e., is $R \in \{0, 1\}^r$.

In Definition 6.2.2, the randomized Turing machine V decides both the position $(Q_1, \ldots, Q_q)$ we're allowed to access, and also a function ϕ_R which only looks at $x_{Q_1}, \ldots, x_{Q_q}$, acting as a decider for V.

Note. Everything is decided before looking at any input.

Just like Cook-Levin theorem is the mother of all *exact hardness*, PCP theorem is the mother of all *hardness of approximation*.

Lecture 20: FGLSS Graph

With Definition 6.2.2, PCP theorem is equivalent to saying the following. 9 Nov. 10:30

Theorem 6.2.4. The PCP theorem is equivalent as saying that there exists $\epsilon > 0$ such that

$$\mathsf{NP} = \mathsf{PCP}_{1,1-\epsilon}(O(\log n), O(1)).$$

Proof. It's easy to see that $\mathsf{NP} \supseteq \mathsf{PCP}_{1,1-\epsilon}(O(\log n), O(1))$ just by considering iterating through all the possible R. Another direction worth a whole class, so we're not going to dive into that. ■

Nevertheless, if we accept that $\mathsf{NP} = \mathsf{PCP}_{1,1-\epsilon}(O(\log n), O(1))$, we can actually show the equivalence between Theorem 6.2.4 and the PCP theorem by showing that Theorem 6.2.4 implies hardness of approximation, specifically, the $(1, 1-\epsilon)$-Gap 3SAT problem.

Firstly, from Cook-Levin theorem, 3SAT is $\mathsf{NP} = \mathsf{PCP}_{1,1-\epsilon}(O(\log n), O(1))$ from assumption. But instead of demonstrate the reduction to 3SAT, we consider max cut instead.

Remark. Generally, between two Gap CSPs with (c_1, s_1) and (c_2, s_2), the hardness is preserve, so we may consider max cut instead since it can be modeled as CSP, and use the machinery to show the hardness for 3SAT.[a]

[a] For more detailed explanation, see Piazza.

Assume $q = 2$, $\psi = \{01, 10\}$, and $r = O(\log n)$. Then there exists V such that given a 3-CNF formula ϕ, it runs in $\mathsf{poly}(|\phi|)$ and only flips r random coins $R \in \{0,1\}^r$, which decides Q_1^R, Q_2^R such that

- if ϕ is satisfiable, $\exists y$ such that $\Pr_R(\psi(y_{Q_1^R}, y_{Q_2^R}) = 1) \geq c$;
- if ϕ is not satisfiable, $\forall y$ such that $\Pr_R(\psi(y_{Q_1^R}, y_{Q_2^R}) = 1) \leq s$.

Notice that the above event $\psi(y_{Q_1^R}, y_{Q_2^R})$ is exactly $y_{Q_1^R} \neq y_{Q_2^R}$. We see that there are at most $2^r \leq n^{O(1)}$ possible R's, and for each R, we access exactly 2 positions, so V will access at most $N := 2 \cdot 2^r$ positions. Now, without loss of generality, we may assume that $\max_R(Q_1^R, Q_2^R) \leq N$.[1]

Consider the optimization problem that finds $y \in \{0,1\}^N$ to maximize the probability of accepting. In this viewpoint, this is just like max cut on $\mathcal{G} = ([N], \{(Q_1^R, Q_2^R) : R \in \{0,1\}^r\})$. Namely, we find a reduction from 3SAT to (c, s)-Gap max cut.

6.3 FGLSS Graph

To see how we utilize PCP theorem, we first see one example.

Problem 6.3.1 (Vertex cover). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, find the smallest $C \subseteq \mathcal{V}$ that covers all $\mathcal{E}$.

Problem 6.3.2 (Independent set). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, find the largest $I \subseteq \mathcal{V}$ that contains no edge.

Problem 6.3.1 and Problem 6.3.2 are often considered together due to the following relation.

Claim. For all $\mathcal{G}$, $\mathsf{OPT}_{\mathrm{VC}}(\mathcal{G}) = |\mathcal{V}| - \mathsf{OPT}_{\mathrm{IS}}(\mathcal{G})$.

Proof. Observe that for all $C \subseteq \mathcal{V}$, C is a vertex cover if and only if $\mathcal{V} \setminus C$ is an independent set. ⊛

6.3.1 Hardness of Vertex Cover and Independent Set

The hardness of vertex cover and independent set can be shown by using the FGLSS graph [Fei+96], which allows us to do reduction from $(1, s)$-Gap 3SAT with $s < 1$, which is NP-hard from the PCP theorem.

[1] Since V is going to access at most N positions anyway, we can just rearrange it.

Consider the input of the $(1, s)$-Gap 3SAT being a 3CNF formula ϕ, n variables $X = \{x_1, \dots, x_n\}$ and $2n$ literals $L = \{x_1, \overline{x}_1, x_2, \overline{x}_2, \dots\}$ with m clauses $\{C_1, \dots, C_m\}$ with three literals in each, i.e., $C_i = (\ell_{i_1} \vee \ell_{i_2} \vee \ell_{i_3})$ with $\ell_{i_j} \in L$.

Then, the goal is to find a reduction from this input to an input (in both cases, it's a graph) of α-Gap vertex cover and β-Gap independent set for some α, β. Toward this goal, we consider the so-called FGLSS graph [Fei+96] $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that

- $\mathcal{V} = [m] \times \left(\{\mathsf{T}, \mathsf{F}\}^3 \setminus (\mathsf{F}, \mathsf{F}, \mathsf{F})\right)$ with $|\mathcal{V}| = 7m$;
- $\left((i, \ell_{i_1}, \ell_{i_2}, \ell_{i_3}), (j, \ell_{j_1}, \ell_{j_2}, \ell_{j_3})\right) \in \mathcal{E}$ if they *contradict*.

The interpretation is that each vertex (i, t_1, t_2, t_3) indicates value of $(\ell_{i_1}, \ell_{i_2}, \ell_{i_3})$, i.e., it's a partial assignment for only 3 variables in C_i.

Notation (Contradiction). If the partial assignment given by two vertices in a FGLSS graph is not consistent, we say they are *contradicting*.

Example. Given $C_1 = (\overline{x}_1 \vee x_2 \vee \overline{x}_3)$ and $C_2 = (x_3 \vee x_4 \vee x_5)$ with two vertices $v = (1, \mathsf{T}, \mathsf{T}, \mathsf{T})$ and $u = (2, \mathsf{T}, \mathsf{F}, \mathsf{F})$, they are contradicting to each other.

Proof. Since v states that $\overline{x}_3$ is T (x_3 is F); while u states that x_3 is T, they contradict. ⊛

This actually finishes the reduction, and the only thing left to do is to determine what α and β is. To do this, observe that following.

Remark. Denote $V_i \coloneqq \{i\} \times \left(\{\mathsf{T}, \mathsf{F}\}^3 \setminus (\mathsf{F}, \mathsf{F}, \mathsf{F})\right)$, we see that V_i is a clique with size 7 since they all contradict to each other.

This means that for independent set, $c = m$; and for vertex cover, $c = |\mathcal{V}| - m = 7m - m = 6m$. We first show the completeness.

Claim. If $\mathsf{OPT}_{\mathrm{3SAT}}(\phi) = 1$, then $\mathsf{OPT}_{\mathrm{IS}}(\mathcal{G}) \geq m$ and $\mathsf{OPT}_{\mathrm{VC}}(\mathcal{G}) \leq 6m$.

Proof. Since ϕ is satisfiable, then there exists $\sigma \colon X \to \{\mathsf{T}, \mathsf{F}\}$ that satisfies every C_i. Then from each V_i, choose a vertex *consistent* with σ.[a]

And since they come from the same assignment σ, there are no contradiction hence no edges between these vertices, i.e., they form a independent set. Hence, $\mathsf{OPT}_{\mathrm{IS}}(\mathcal{G}) \geq m$, and $\mathsf{OPT}_{\mathrm{VC}}(\mathcal{G}) \leq 6m$. ⊛

[a]There are exactly one for each i.

We now show the soundness. In particular, we will always deal with contrapositive in this course, i.e., instead of find a bad input from a bad input, we find a good input from a good input, but backwards.

Claim. If $\mathsf{OPT}_{\mathrm{3SAT}}(\phi) < s$, then $\mathsf{OPT}_{\mathrm{IS}}(\mathcal{G}) < sm$ and $\mathsf{OPT}_{\mathrm{VC}}(\mathcal{G}) > (7 - s)m$.

Proof. Consider the contrapositive, i.e., we show that $\mathsf{OPT}_{\mathrm{IS}}(\mathcal{G}) \geq sm$ (hence $\mathsf{OPT}_{\mathrm{VC}}(\mathcal{G}) \leq (7 - s)m$), then $\mathsf{OPT}_{\mathrm{3SAT}}(\phi) \geq s$.

Let $I \subseteq \mathcal{V}$ be an independent set such that $|I| \geq sm$, and let $\sigma \colon X \to \{\mathsf{T}, \mathsf{F}\}$ such that for all C_i with $|I \cap V_i| = 1$,[a] assign variables in C_i according to $I \cap V_i$.[b] Finally, we extend it arbitrarily for unassigned variables if needed. We see that for all C_i such that $|I \cap V_i| = 1$, this assignment σ satisfies C_i, hence σ satisfies exactly $|I| \geq sm$ clauses, i.e., the normalized optimal solution for 3SAT is $\geq sm/m = s$ as required. ⊛

[a]It can only be the case that I doesn't include vertices from some V_i, but if it does, no more than 1 can be included since V_i is a clique.

[b]This is well-defined since there are no contradictions with I being an independent set.

With the above discussion, we see that the Gap is $\beta = s < 1$, $\alpha = (7 - s)/6 > 1$. Hence, there exists a reduction from $(1, s)$-Gap SAT to $(7 - s)/6$-Gap vertex cover and s-Gap independent set.

Remark. Actually, it's also easy to check that there exists a reduction from (c, s)-Gap P to $(f - s)/(f - c)$-Gap vertex cover and s/c-Gap independent set for any CSP P with f being the number of satisfying assignments.

From this, we have the following.

Theorem 6.3.1. For all $\epsilon > 0$, there exists a CSP P such that $(1, \epsilon)$-Gap P is NP-hard.

Corollary 6.3.1. For all $c > 0$, there exists no c-approximation algorithm for independent set.

The state-of-the-art in-approximation result for independent set result is the following.

Theorem 6.3.2. For all $\epsilon > 0$, there exists no $1/n^{1-\epsilon}$-approximation algorithm for independent set.

6.4 Label Cover

Although PCP theorem is powerful as we just saw, but there is also another useful problem to study when doing reduction, the label cover.

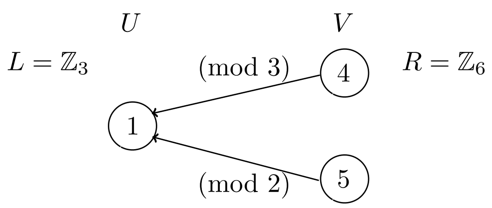
Problem 6.4.1 (Label cover). Given a d-regular bipartite graph $\mathcal{G} = (U \sqcup V, \mathcal{E})$ with $|U| = |V| = n$, with label sets L (for U) and R (for V) with $|R| \geq |L|$ such that for all $e = (u, v) \in \mathcal{E}$, we have a projection $\Pi_e : [R] \to [L]$. The *label cover* problem asks for an assignment $\sigma : U \sqcup V \to L \cup R$ such that

$$\sigma|_U : U \to L, \quad \sigma|_V : V \to R,$$

maximizes the number of satisfied edge.[a]

[a] The edge $e = (u, v)$ is satisfied by σ if $\Pi_e(\sigma(v)) = \sigma(u)$.

Note. Notice that d-regularity directly implies that $|U| = |V|$.



For label cover, the parameters are $|U|, |V|, |\mathcal{E}|, L, R$, and we sometimes for simplicity, use L and R to also denote the size of L and R.

Remark (Baseline). There is a trivial $1/L$-approximation algorithm.

Proof. Consider a random assignment σ such that

- for all $v \in V$, $\sigma(v)$ randomly from $[R]$;
- for all $u \in U$, $\sigma(u)$ randomly from $[L]$.

Fix $e = (u, v)$, we see that $\Pr(\Pi_e(\sigma(v)) = \sigma(u)) = 1/L$. ⊛

Theorem 6.4.1. For all $\epsilon > 0$, there exists L, R such that the $(1, \epsilon)$-Gap label cover for L, R is NP-hard.

Proof. This is based on the PCP theorem with parallel repetition theorem. ∎

6.4.1 Hardness of Max k-Coverage

Recall the max k-coverage problem.

Problem 6.4.2 (Max k-coverage)**.** Given a set system $(\Omega, \mathcal{S})$ and k, finds $\mathcal{S}' \subseteq \mathcal{S}$ such that $|\mathcal{S}|' = k$ which maximizes $|\bigcup_{S \in \mathcal{S}'} S|/|\Omega|$.

We're going to see the hardness of the max k-coverage problem, and our goal is to prove the following.

Theorem 6.4.2. For all $\epsilon > 0$, there is no $(3/4 + \epsilon)$-approximation algorithm for max k-coverage.

Interestingly, the state-of-the-art result is the following.

Theorem 6.4.3. For all $\epsilon > 0$, there is no $(1 - 1/e + \epsilon)$-approximation algorithm for max k-coverage.

By proving Theorem 6.4.2, we almost prove Theorem 6.4.3!

Lecture 21: Reduction to Max k-Coverage

We now see the reduction from $(1, \epsilon_0)$-Gap label cover to $(1, 3/4 + \epsilon)$-Gap max k-coverage and use Theorem 6.4.1 to prove Theorem 6.4.2. Specifically, we'll show that $\epsilon_0 = \epsilon^3/2000$. Given a label cover instance $\mathcal{G} = (U \sqcup V, \mathcal{E})$, L, R and $\{\Pi_e\}_{e \in \mathcal{E}}$, consider 14 Nov. 10:30

$$\Omega \coloneqq \mathcal{E} \times \{0,1\}^L$$

such that $(e, x_1, \dots, x_L) \in \Omega$ with $|\Omega| = |\mathcal{E}| \cdot 2^L$. Then, the reduction is given by

- For all $u \in U$, $i \in [L]$, $S_{u,i} = \{(e, x_1, \dots, x_L) \colon e \ni u, x_i = 0\}$.
- For all $v \in V$, $i \in [R]$, $S_{v,i} = \left\{(e, x_1, \dots, x_L) \colon e \ni v, x_{\Pi_e(i)} = 1\right\}$.
- $k = 2n = |U| + |V|$.

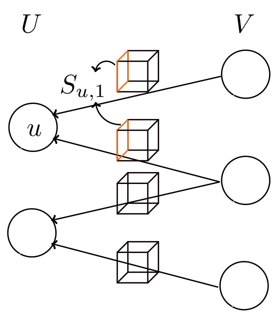


Figure 6.1: u and $S_{u,1}$.

We first show the completeness, where we want to show that if $\mathsf{OPT}_{\text{M-}k\text{-C}} = 1$, then $\mathsf{OPT}_{\text{LC}} = 1$. Given an accepted (perfect) instance of label cover, there exists σ such that for all $e = (u, v)$, $\sigma(u) = \Pi_e(\sigma(v))$. Then, we construct

$$\mathcal{S}' \coloneqq \left\{S_{u,\sigma(u)}\right\}_{u \in U} \cup \left\{S_{v,\sigma(v)}\right\}_{v \in V}.$$

Indeed, $\mathcal{S}'$ covers every element in Ω since for all $(e, x) \in \Omega$,

$$\begin{cases} (e,x) = ((u,v),x) \in S_{u,\sigma(u)}, & \text{if } x_{\sigma(u)} = 0; \\ (e,x) = ((u,v),x) \in S_{v,\sigma(v)}, & \text{if } x_{\sigma(u)} = 1, \end{cases}$$

where the later one is from $\Pi_e(\sigma(v)) = \sigma(u)$.

To prove soundness, we consider the contrapositive, namely if $\mathsf{OPT}_{\text{M-}k\text{-C}} \geq (3/4 + \epsilon)$, then $\mathsf{OPT}_{\text{LC}} \geq \epsilon_0$. To start with, assume that there exists $\mathcal{S}'$ such that $|\mathcal{S}'| = k = 2n$ which covers at least $(3/4 + \epsilon)$ fraction of Ω.

(a) Suppose for all $u \in U$, $|\mathcal{S}' \cap \{S_{u,i} : i \in [L]\}| = 1$ and for all $v \in V$, $|\mathcal{S}' \cap \{S_{v,i} : i \in [R]\}| = 1$.[2] Then we let σ be the labeling which is consistent with $\mathcal{S}'$. This is indeed a good solution since for every $e = (u, v) \in \mathcal{E}$, $S_{u,\sigma(u)}$ and $S_{v,\sigma(v)}$ cover

$$\begin{cases} 1, & \text{if } \Pi_e(\sigma(v)) = \sigma(u); \\ 3/4, & \text{otherwise} \end{cases}$$

fraction of C_e, where C_e is the hypercube corresponding to e.[3] This is because if (e, x) is not covered, then $x_{\sigma(u)} = 1$ and $x_{\Pi_e(\sigma(v))} = 0$, which is exactly $1/4$ of C_e. Hence,

$$\underbrace{\overbrace{\text{fraction of elements satisfied by } \sigma}^{a} \cdot 1 + \overbrace{\text{fraction of elements unsatisfied by } \sigma}^{1-a} \cdot \frac{3}{4}}_{\text{fraction of elements covered by } \mathcal{S}'} \geq \frac{3}{4} + \epsilon$$

from the assumption. Then, $a + (1 - a) \cdot 3/4 \geq 3/4 + \epsilon$, which implies $a \geq 4\epsilon \geq \epsilon^3/2000$.

Problem. Compared to this warm-up case, $\mathcal{S}'$ can have many sets from some u, v and none from others.

(b) For all $u \in U$, let $\ell(u) \coloneqq \{i \in L : S_{u,i} \in \mathcal{S}'\}$, and for all $v \in V$, let $\ell(v) \coloneqq \{i \in R : S_{v,i} \in \mathcal{S}'\}$. Then,

$$\mathbb{E}_{v \in U \sqcup V}\left[|\ell(v)|\right] = 1$$

since there are $k = 2n$ labels, and we have exactly $k = 2n$ sets in $\mathcal{S}$. Hence, since $|\mathcal{E}| = nd$ from d-regularity,

$$\begin{aligned} \mathbb{E}_{e=(u,v)\in\mathcal{E}}\left[|\ell(u)| + |\ell(v)|\right] &= \frac{1}{nd} \sum_{e=(u,v)\in\mathcal{E}} \left[|\ell(u)| + |\ell(v)|\right] \\ &= \frac{1}{nd} \cdot d \left(\sum_{u\in V} |\ell(u)| + \sum_{v \in V} |\ell(v)| \right) = 2 \end{aligned}$$

Intuition. We see that in expectation, this general case is same as the first warm-up case.

Now, to construct a label cover σ, we define for all u, $\sigma(u)$ be a random element from $\ell(u)$, and nothing if $\ell(u) = \varnothing$. We say $e = (u, v)$ is consistent if $\Pr_\sigma(e \text{ is satisfied}) > 0$, which is equivalent to say $\ell(u) \cap \Pi_e(\ell(v)) \neq \varnothing$.

Claim. If e is not consistent, then $\mathcal{S}'$ covers

$$1 - 2^{-(|\ell(u)| + |\Pi(\ell(v))|)} \leq 1 - 2^{-(|\ell(u)| + |\ell(v)|)}$$

fraction of C_e.

Proof. Without loss of generality, let $\ell(u) = \{1, \ldots, a\}$, and $\Pi(\ell(v)) = \{a + 1, \ldots, a + b\}$. Then for $x \in \{0, 1\}^L$, (e, x) is covered if and only if $x_i = 0$ for some $i \in \{1, \ldots, a\}$ and $x_j = 1$ for some $j \in \{a + 1, \ldots, a + b\}$. Hence, exactly $1 - 2^{-(a+b)}$ fraction of elements in C_e are covered. ⊛

Finally, we say $e = (u, v)$ is frugal if $|\ell(u)| + |\ell(v)| \leq 10/\epsilon$, and is good if e is both consistent and frugal. Then, we see that

$$\Pr(e \text{ is satisfied}) \geq \frac{\epsilon^2}{100}.$$

Then if $\epsilon/20$ fraction of edges is good, then the fraction of satisfied edges is larger than

$$\frac{\epsilon}{20} \cdot \frac{\epsilon^2}{100} = \frac{\epsilon^3}{2000} = \epsilon_0.$$

So, now we just need to show that there are actually $\epsilon/20$ fraction of edges is good.

[2] This case is a warm-up case such that one vertex can only choose one set.

[3] I.e., $C_e \coloneqq \left\{ x \in \{0, 1\}^L : (e, x) \in \Omega \right\}$.

Claim. At least $\epsilon/20$ fraction of edges is good.

Proof. Assume otherwise. Then

$$\begin{aligned}\Pr_e(e \text{ is consistent}) &= \Pr(e \text{ is good}) + \Pr(e \text{ is consistent but not frugal}) \\ &\leq \underbrace{\Pr(e \text{ is good})}_{\leq \epsilon/20} + \underbrace{\Pr(e \text{ is not frugal})}_{\leq 2\epsilon/10} \\ &\leq \frac{\epsilon}{4},\end{aligned}$$

where the bound follows from the Markov inequality.[a] Finally, we see that

$$\begin{aligned}2 &= \mathbb{E}_{e=(u,v)}\left[|\ell(u)| + |\ell(v)|\right] \\ &= \Pr(e \text{ is consistent}) \cdot \mathbb{E}\left[|\ell(u)| + |\ell(v)| \mid e \text{ is consistent}\right] \\ &\quad + \Pr(e \text{ is not consistent}) \cdot \mathbb{E}\left[|\ell(u)| + |\ell(v)| \mid e \text{ is not consistent}\right] \\ &=: \Pr(e \text{ is consistent}) \cdot a + \Pr(e \text{ is not consistent}) \cdot b.\end{aligned}$$

Since $a \geq 2$ from the fact that for e being consistent, we need at least two labels, so $b \leq 2$ since the overall expectation is 2. Now, define r_e to be

$$r_e = \frac{\left|\bigcup_{S \in \mathcal{S}'} (S \cap C_e)\right|}{|C_e|},$$

i.e., the fraction of elements in C_e covered by $\mathcal{S}'$. Then,

$$\begin{aligned}\mathbb{E}_{e=(u,v)}\left[r_e \mid e \text{ is not consistent}\right] &\leq \mathbb{E}_{e=(u,v)}\left[1 - 2^{-|\ell(u)|+|\ell(v)|} \mid e \text{ is not consistent}\right] \\ &\leq 1 - 2^{-\mathbb{E}_{e=(u,v)}[|\ell(u)|+|\ell(v)| \mid e \text{ is not consistent}]} \leq 1 - 2^2 = \frac{3}{4},\end{aligned}$$

from the Jensen's inequality since -2^{-x} is a concave function. Hence,

$$\begin{aligned}\mathbb{E}_{e\in\mathcal{E}}\left[r_e\right] &= \mathbb{E}_{e\in\mathcal{E}}\left[r_e \mid e \text{ is consistent}\right] \cdot \Pr(e \text{ is consistent}) \\ &\quad + \mathbb{E}_{e\in\mathcal{E}}\left[r_e \mid e \text{ is not consistent}\right] \cdot \Pr(e \text{ is not consistent}) \leq \frac{\epsilon}{4} + \frac{3}{4} < \frac{3}{4} + \epsilon,\end{aligned}$$

which is contradiction since we assume $\mathcal{S}'$ covers at least $3/4 + \epsilon$ fraction of elements. ⊛

[a] If not true, then $\mathbb{E}_e\left[|\ell(u)| + |\ell(v)|\right] > 2\epsilon/10 \cdot 10/\epsilon \geq 2$, which contradicts to what we have shown.

In all, we see that Theorem 6.4.2 is proved with Theorem 6.4.1 since we have a valid reduction.

Chapter 7

Unique Games and the Conjecture

Lecture 22: 3LIN and BLR Test

In this section, we're going to study a special problem of label cover, called the unique games. As we have already seen, the hardness of label cover already implies the hardness of problems like max k-coverage, as shown in Theorem 6.4.2. 16 Nov. 10:30

In particular, if we assume a hardness conjecture called unique game conjecture, we can show the optimal hardness for 3LIN, 3SAT, and max cut. Let's first consider the first two.

7.1 Optimal Hardness for 3LIN and 3SAT

Let consider the following problem.

Problem 7.1.1 (MAX-3LIN). Given $X \coloneqq \{x_1, \ldots, x_n\}$, $\Sigma = \mathbb{F}_2$, and a set of m equations in the form of $x_i + x_j + x_k = 0$ or 1. The problem *MAX-3LIN* asks to find $\sigma \colon X \to \mathbb{F}_2$ that maximizes the fraction of the satisfied equations.

Remark. We often call MAX-3LIN as 3LIN for brevity.

We're going to show the hardness of 3LIN, but first, note the following.

Claim. $(1, 1)$-Gap 3LIN can be solved in polynomial time.

Proof. Consider solving

$$\begin{bmatrix} 1 & 0 & \dots & 1 & 1 \\ 1 & 1 & \dots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & \dots & 0 & 0 \end{bmatrix}_{m\times n} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_{n\times 1} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}_{m\times 1},$$

where the coefficient matrix has only three non-zero entries for each row. Then, if this is an accepted instance, this system of equations has a solution, and we can check this just by Gaussian elimination over $\mathbb{F}_2$. ⊛

But what if the instance has $\mathsf{OPT} = 1 - \epsilon$ for some small $\epsilon > 0$? In this case, this question is actually hard.

Remark. A trivial approximation for 3LIN is just to do a random assignment, which gives us a $1/2$-approximation.

The hardness result of 3LIN we're going see is the following.

Theorem 7.1.1 ([Hås97]). For every constant $\epsilon > 0$, the $(1-\epsilon, 1/2+\epsilon)$-Gap 3LIN is NP-hard.

Which implies the following.

Corollary 7.1.1. For every constant $\epsilon > 0$, the $(1-\epsilon, 7/8+\epsilon)$-Gap 3SAT is NP-hard.

Proof. Given 3LIN instance, we create an 3SAT instance

$$(x_i + x_j + x_k = 0) \Rightarrow \begin{cases} (\overline{x}_i \vee x_j \vee x_k); \\ (x_i \vee \overline{x}_j \vee x_k); \\ (x_i \vee x_j \vee \overline{x}_k); \\ (\overline{x}_i \vee \overline{x}_j \vee \overline{x}_k); \end{cases}, \quad (x_i + x_j + x_k = 1) \Rightarrow \begin{cases} (\overline{x}_i \vee \overline{x}_j \vee x_k); \\ (x_i \vee \overline{x}_j \vee \overline{x}_k); \\ (\overline{x}_i \vee x_j \vee \overline{x}_k); \\ (x_i \vee x_j \vee x_k). \end{cases}$$

We see that

- a 3LIN equation is satisfied $\Leftrightarrow$ 4 corresponding 3SAT clauses are satisfied;
- a 3LIN equation is unsatisfied $\Leftrightarrow$ 3 corresponding 3SAT clauses are satisfied.

So,

$$\mathsf{OPT}_{\text{3SAT}} = \mathsf{OPT}_{\text{3LIN}} \cdot \frac{4}{4} + (1 - \mathsf{OPT}_{\text{3LIN}}) \cdot \frac{3}{4} = \frac{3}{4} + \frac{1}{4} \cdot \mathsf{OPT}_{\text{3LIN}},$$

and hence the $(1-\epsilon, 1/2+\epsilon)$-Gap hardness for 3LIN from Theorem 7.1.1 implies the $(1-\epsilon, 7/8+\epsilon)$-Gap hardness for 3SAT. ∎

Remark. Actually, with more work, $(1, 7/8+\epsilon)$-Gap 3SAT is also NP-hard.

Note. Recall that a random assignment (such as Algorithm A.1) of 3SAT satisfies 7/8 fraction of clauses from Lemma A.1.1. This suggests that both 3SAT and 3LIN is hard: we can't do better than random assignment.

So, we will embark a long journey to prove Theorem 7.1.1 from label cover, i.e., we again want to find a good assignment $\sigma \colon U \sqcup V \to L \sqcup R$ by using the hypercube construction. To do this, we need to study Fourier analysis over $\{\pm 1\}^n$ of a boolean function.

7.2 Fourier Analysis of Boolean Functions

We now introduce a powerful tool which is well-known in engineering, Fourier analysis. While it's powerful for infinite-domain function classes, it's not that well-known in the class of finite-domain functions. However, in the latter case, it turns out to be still powerful.

7.2.1 Boolean Functions and Boolean-Valued Functions

Firstly, we introduce the boolean function.

Definition. Let $\mathbb{F}_2$ be the additive group over $\mathbb{F}_2 = \{0, 1\}$ and consider the conical isomorphism to the multiplicative group $\{\pm 1\}$.

Definition 7.2.1 (Boolean funciton). A function f is a *boolean function* if $f \colon \{\pm 1\}^n \to \mathbb{R}$.

Definition 7.2.2 (Boolean-valued). If the range of a boolean function f is $\{\pm 1\}$, we say f is a *boolean-valued* function.

Note. Since the domain of a boolean function has cardinality 2^n, we can identify it as a 2^n-dimensional vector.

Consider viewing the set of boolean functions as a Hilbert space, we then define the following inner product between f, g as

$$\langle f, g\rangle = \frac{1}{2^n} \sum_{x\in\{\pm 1\}^n} f(x)g(x) =: \mathbb{E}_x\left[f(x)g(x)\right].$$

Note. We have $\|f\|_2^2 = \langle f, f\rangle = \mathbb{E}_x\left[f(x)^2\right]$.

Now, we want to know what are the orthonormal basis for the set of boolean functions. There are two important examples:

(a) Standard basis: For all $x \in \{\pm 1\}^n$,

$$f_x\colon \{\pm 1\}^n \to \mathbb{R}, \quad f_x(y) = \begin{cases} 2^{n/2}, & \text{if } x = y; \\ 0, & \text{otherwise.}\end{cases}$$

We see that $\{f_x\}_{x\in\{\pm 1\}^n}$ is an orthonormal basis since for all x, $\|f_x\|_2^2 = 1$ and $\langle f_x, f_y\rangle = 0$ for all $x \neq y$.

(b) Fourier basis: For all $S \subseteq [n]$, define $\chi_\varnothing(x) \coloneqq 1$ and

$$\chi_S\colon \{\pm 1\}^n \to \{\pm 1\}, \quad \chi_S(x) = \prod_{i\in S} x_i =: x^S.$$

We see that $\{\chi_S\}_{S\subseteq[n]}$ is an orthonormal basis since $\mathbb{E}_x\left[\chi_S(x)^2\right] = 1$, and

$$\langle \chi_S, \chi_T\rangle = \mathbb{E}_x\left[\chi_S(x)\chi_T(x)\right] = \mathbb{E}_x\left[x^S x^T\right] = \mathbb{E}_x\left[x^{S\Delta T}\right] = \begin{cases} 0, & \text{if } S \neq T; \\ 1, & \text{if } S = T.\end{cases}$$

7.2.2 Fourier Analysis over Boolean Functions

We'll study the Fourier basis primarily. Firstly, we have the following decomposition of $f\colon \{\pm 1\}^n \to \mathbb{R}$ as

$$f = \sum_{S\subseteq[n]} \hat{f}(S)\chi_S,$$

where we call $\hat{f}(S)$ the Fourier coefficient. Now, here is some basic facts and theorem.

Proposition 7.2.1. Given an orthonormal basis $\{\chi_S\}_{S\subseteq[n]}$ of the space of boolean functions, then $\hat{f}(S) = \langle f, \chi_S\rangle$.

Proof. Since $\langle f, \chi_S\rangle = \left\langle \sum_{T\subseteq[n]} \hat{f}(T)\chi_T, \chi_S\right\rangle = \sum_T \hat{f}(T)\langle \chi_T, \chi_S\rangle = \hat{f}(S)$. ■

Theorem 7.2.1 (Plancherel's theorem)**.** Given two boolean functions f, g, we have

$$\langle f, g\rangle = \mathbb{E}_x\left[f(x)g(x)\right] = \sum_{S\subseteq[n]} \hat{f}(S)\hat{g}(S).$$

Proof. Since $\langle f, g\rangle = \left\langle \sum_S \hat{f}(S)\chi_S, \sum_T \hat{f}(T)\chi_T\right\rangle = \sum_{S,T} \hat{f}(S)\hat{g}(T)\langle \chi_S, \chi_T\rangle = \sum_S \hat{f}(S)\hat{g}(S)$. ■

Theorem 7.2.2 (Parseval's theorem)**.** Given a boolean function f, $\|f\|_2^2 = \sum_{S\subseteq[n]} \hat{f}(S)^2$.

Proof. This directly follows from Plancherel's theorem with $\|f\|_2^2 = \langle f, f\rangle$. ■

Claim. Given a boolean function f, $\mathbb{E}_x\left[f(x)\right] = \hat{f}(\varnothing)$.

Proof. Since $\mathbb{E}_x[f(x)] = \mathbb{E}_x[f(x) \cdot 1] = \mathbb{E}_x[\chi_\varnothing(x) f(x)] = \langle \chi_\varnothing, f \rangle = \hat{f}(\varnothing)$. ⊛

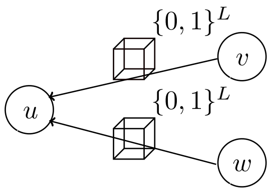
Claim. Given a boolean function f, $\operatorname{Var}[f] = \sum_{S \neq \varnothing} \hat{f}(S)^2$.

Proof. Since $\operatorname{Var}[f] = \mathbb{E}_x\left[f^2\right] - (\mathbb{E}_x[f])^2 = \sum_{S \subseteq [n]} \hat{f}(S)^2 - \hat{f}(\varnothing)^2 = \sum_{S \neq \varnothing} \hat{f}(S)^2$. ⊛

7.3 BLR Test and the Noisy BLR Test

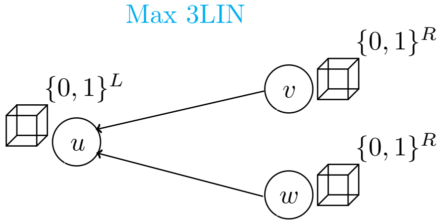
Our goal is to show the reduction from label cover to 3LIN by designing a 3LIN instance on variables $U \times \{\pm 1\}^L$ identified by hypercube. Compared to the max k-coverage, now the hypercubes are variables.

Max k-Coverage



for each hypercube, how it is covered
is determined by $\sigma(u), \sigma(v)$

Max 3LIN



for each hypercube, how it is assigned
is determined by $\sigma(u)$

Ideally, (u, x) gets $x_{\sigma(u)}$, which is just a particular base of the Fourier basis called dictation.

Definition 7.3.1 (Dictation). The *dictation function* of i is defined as $\chi_i(x) \coloneqq \chi_{\{i\}}(x) = x_i$.

For convenience, we sometimes call a dictation function as *dictator* for short.

Intuition. Consider assignment $\alpha\colon \{\text{variables of 3LIN}\} = U \times \{\pm 1\}^L \to \{\pm 1\}$, then fix $u \in U$, and let $f\colon \{\pm 1\}^L \to \{\pm 1\}$ given by $f(x) = \alpha(u, x)$. Ideally, $f(x) = \chi_{\sigma(u)}$, which means $f \in \{\chi_1, \ldots, \chi_L\}$.

Problem. But since the number of possible f's is 2^{2^L}, how can we *force* f to be in $\{\chi_i\}_{i=1}^L$, or even $\{\chi_S\}_{S \subseteq [L]}$?

Answer. We can design a 3LIN instance on variables $\{\pm 1\}^n$ such that if the *assignment* $f\colon \{\pm 1\}^n \to \{\pm 1\}$ is *close* to some χ_S, $\operatorname{obj}(f)$[a] is large; if it is *far* from any χ_S, then $\operatorname{obj}(f)$ is small. ⊛

[a]The $\operatorname{obj}(f)$ is the fraction of equations satisfied by f.

Lecture 23: Noisy BLR Test and Unique Games Conjecture

7.3.1 BLR Test

21 Nov. 10:30

To do this, we consider creating a *weighted* 3LIN instance.

(a) Each equation has weight.

(b) Weights sum to 1.

(c) $\operatorname{obj}(f) = \sum_{C_i \text{ is satisfied}} w(C_i)$.

In this way, we can interpret one instance as the probability distribution over equations. Rather than construct the equation directly, we can specify how we are going to sample equations via specifying a probability distribution!

Remark. Mathematically, given a distribution P on $\{x_i \cdot x_j \cdot x_k = b\}$, we have

$$w(w_i \cdot w_j \cdot w_k = b) = \Pr_P(x_i \cdot x_j \cdot x_k = b \text{ is sampled}).$$

Now, the construction is the following, where we specify how we sample one equation (hence this gives the probability distribution).

- Variables: $\{\pm 1\}^n$ (hence, the assignment is a boolean function $f\colon \{\pm 1\}^n \to \{\pm 1\}$).
- Sample $x \in \{\pm 1\}^n$ and $y \in \{\pm 1\}^n$ independently, let $z = \langle x, y \rangle$.
- Output $f(x) \cdot f(y) \cdot f(z) = 1$.

Note. Notice that this construction is deterministic, i.e., we can specify the weight of each possible equations directly. But we just use the probabilistic language.

To see why this is what we want, notice that if $f = \chi_S$ for some $S \subseteq [n]$, then for all $x, y \in \{\pm 1\}^n$,

$$f(x)f(y)f(z) = x^S y^S (x \cdot y)^S = 1,$$

hence $\operatorname{obj}(f) = 1$! On the other hand, if f has k nonzero Fourier coefficients (i.e., f is far from any χ_S) of value $1/\sqrt{k}$, then $\operatorname{obj}(f) \approx 1/2$. This construction is known as BLR test [BLR90].

Theorem 7.3.1 ([BLR90]). Under the construction, $\operatorname{obj}(f) = 1/2 + 1/2 \cdot \sum_{S \subseteq [n]} \hat{f}(S)^3$.

Proof. Observe that

$$\begin{aligned}
\operatorname{obj}(f) &= \mathbb{E}_{x,y}\left[\mathbb{1}\left[f(x)f(y)f(z) = 1\right]\right] \\
&= \mathbb{E}_{x,y}\left[\frac{1}{2} + \frac{1}{2} f(x)f(y)f(z) = 1\right] = \frac{1}{2} + \frac{1}{2}\mathbb{E}_{x,y}\left[f(x)f(y)f(z)\right].
\end{aligned}$$

Now, we decompose $f(x), f(y), f(z)$ with respect to the Fourier basis, i.e., $f(x) = \sum_S \hat{f}(s)\chi_S(x)$ we have

$$\begin{aligned}
\operatorname{obj}(f) &= \frac{1}{2} + \frac{1}{2}\mathbb{E}_{x,y}\left[\sum_{S,T,U \subseteq [n]} \hat{f}(S)\hat{f}(T)\hat{f}(U) \cdot \chi_S(x)\chi_T(y)\chi_U(z)\right] \\
&= \frac{1}{2} + \frac{1}{2} \sum_{S,T,U \subseteq [n]} \hat{f}(S)\hat{f}(T)\hat{f}(U) \cdot \mathbb{E}_{x,y}\left[\chi_S(x)\chi_T(y)\chi_U(z)\right].
\end{aligned}$$

Since for fixed S, T, U,

$$\begin{aligned}
\mathbb{E}_{x,y}\left[\chi_S(x)\chi_T(y)\chi_U(z)\right] &= \mathbb{E}_{x,y}\left[x^S \cdot y^T \cdot (x \cdot y)^U\right] \\
&= \mathbb{E}_{x,y}\left[x^{S\Delta U} y^{T;\Delta U}\right] \\
&= \mathbb{E}_x\left[x^{S\Delta U}\right]\mathbb{E}_y\left[y^{T\Delta U}\right] = \begin{cases} 1, & \text{if } S = U \text{ and } T = U; \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}$$

Hence,

$$\operatorname{obj}(f) = \frac{1}{2} + \frac{1}{2} \sum_{S \subseteq [n]} \hat{f}(S)^3.$$

■

We see that if we only require f to have high value when it is corresponding to χ_S, then we're done. But actually, what we want is when S is a singleton set, and hence we need more works.

7.3.2 Noisy BLR Test

In particular, we want to eliminate the case that when $S = \varnothing$ and $[n]$, $\mathrm{obj}(f) = 1$, i.e., we want to implement dictation test. This can be done via introducing noise. Firstly, consider the following new 3LIN instances.

- Variables: $\{\pm 1\}^n$ (hence, the assignment is a boolean function $f\colon \{\pm 1\}^n \to \{\pm 1\}$).
- Sample $x \in \{\pm 1\}^n$ and $y \in \{\pm 1\}^n$ independently, and also $b \in \{\pm 1\}$.
- For all i, let
$$z_i = \begin{cases} x_i y_i b, & \text{with probability } 1-\epsilon; \\ -x_i y_i b, & \text{with probability} \epsilon. \end{cases}$$
- Output $f(x) \cdot f(y) \cdot f(z) = b$.

Remark (Sanity check). If $f = \chi_\varnothing$, $\mathrm{obj}(f) = 1/2$; if $f = \chi_{[n]}$, $\mathrm{obj}(f) \approx 1/2$.

Proof. We see that

- If $f = \chi_\varnothing$ (i.e., $f(x) = 1$ for all x): $\mathrm{obj}(f) = 1/2$.
- If $f = \chi_{[n]}$ (i.e., $f(x) = x_1 x_2 \dots x_n$): $\mathrm{obj}(f) = \mathbb{E}_{x,y,z,b}\left[1/2 + 1/2 f(x)f(y)f(z)b\right]$. Since

$$\begin{aligned}\mathbb{E}\left[f(x)f(y)f(z)b\right] &= \frac{1}{2}\mathbb{E}\left[f(x)f(y)f(z)b \mid b=1\right] + \frac{1}{2}\mathbb{E}\left[f(x)f(y)f(z)b \mid b=-1\right] \\ &= \frac{1}{2}\prod_{i=1}^n \mathbb{E}\left[x_i y_i z_i \mid b=1\right] - \frac{1}{2}\prod_{i=1}^n \mathbb{E}\left[x_i y_i z_i \mid b=-1\right] \\ &= \frac{1}{2}(1-2\epsilon)^n - \frac{1}{2}(-1+2\epsilon)^n \\ &= \begin{cases} 0, & \text{if } n \text{ is even;} \\ (1-2\epsilon)^n, & \text{if } n \text{ is odd.} \end{cases}\end{aligned}$$

So, if $n \gg 1/\epsilon$, then $(1-2\epsilon)^n \leq e^{-2\epsilon n} \approx 0$, i.e., $\mathrm{obj}(f) \approx 1/2$.

⊛

The above remark holds for a general f, i.e., when f is far from dictation, the value will be less than 1 significantly. To see this, let $f(x) = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S$, recall that

$$\mathrm{obj}(f) = \mathbb{E}_{x,y,z,b}\left[\frac{1}{2} + \frac{1}{2}f(x)f(y)f(z)b\right] = \frac{1}{2} + \frac{1}{2}\sum_{S,T,U \subseteq [n]} \hat{f}(S)\hat{f}(T)\hat{f}(U)\mathbb{E}\left[\chi_S(x)\chi_T(y)\chi_U(z)b\right],$$

hence we're interested in

$$\mathbb{E}_{x,y,z,b}\left[\chi_S(x)\chi_T(y)\chi_U(z)b\right] = \frac{1}{2}\mathbb{E}\left[\chi_S(x)\chi_T(y)\chi_U(z) \mid b=1\right] - \frac{1}{2}\mathbb{E}\left[\chi_S(x)\chi_T(y)\chi_U(z) \mid b=-1\right]$$

for a fixed $S, T, U \subseteq [n]$. Notice that in this expectation, things are independent among the coordinate, i.e., if S contains i, then x_i will appear in the calculation, and same for T and U. But observe the following.

Claim. For all $i \in [n]$, given $b = \pm 1$,

$$\mathbb{E}\left[x_i\right] = \mathbb{E}\left[y_i\right] = \mathbb{E}\left[z_i\right] = \mathbb{E}\left[x_i y_i\right] = \mathbb{E}\left[x_i z_i\right] = \mathbb{E}\left[y_i z_i\right] = 0$$

Proof. Consider the case that $b = 1$, since

$$z_i = \begin{cases} x_i y_i, & \text{w.p. } 1-\epsilon; \\ -x_i y_i, & \text{w.p. } \epsilon; \end{cases}, \quad x_i z_i = \begin{cases} y_i, & \text{w.p. } 1-\epsilon; \\ -y_i, & \text{w.p. } \epsilon, \end{cases}$$

we have that $\mathbb{E}[x_i z_i] = (1-\epsilon)\mathbb{E}[y_i] - \epsilon\mathbb{E}[y_i] = 0$. The same holds for $b = -1$ as well. ⊛

And hence, we see that only $\mathbb{E}[x_i y_i z_i]$ is left, with

$$\mathbb{E}[x_i y_i z_i \mid b = 1] = 1 - 2\epsilon, \quad \mathbb{E}[x_i y_i z_i \mid b = -1] = -1 + 2\epsilon.$$

This suggests that

$$\mathbb{E}[\chi_S(x)\chi_T(y)\chi_U(z) \mid b = 1] = \begin{cases} (1-2\epsilon)^{|S|}, & \text{if } S = T = U; \\ 0, & \text{otherwise.} \end{cases}$$
$$\mathbb{E}[\chi_S(x)\chi_T(y)\chi_U(z) \mid b = -1] = \begin{cases} (-1+2\epsilon)^{|S|}, & \text{if } S = T = U; \\ 0, & \text{otherwise.} \end{cases}$$

And hence,

$$\mathbb{E}_{x,y,z,b}[\chi_S(x)\chi_T(y)\chi_U(z)b] = \begin{cases} (1-2\epsilon)^{|S|}, & \text{if } S = T = U \text{ and } |S| \text{ is odd;} \\ 0, & \text{otherwise,} \end{cases}$$

implying that

$$\begin{aligned} \operatorname{obj}(f) &= \mathbb{E}_{x,y,z,b}\left[\frac{1}{2} + \frac{1}{2} f(x)f(y)f(z)b\right] \\ &= \frac{1}{2} + \frac{1}{2}\sum_{S,T,U\subseteq[n]} \hat{f}(S)\hat{f}(T)\hat{f}(U)\mathbb{E}[\chi_S(x)\chi_T(y)\chi_U(z)b] \\ &= \frac{1}{2} + \frac{1}{2}\sum_{|S| \text{ odd}} \hat{f}(S)^3(1-2\epsilon)^{|S|}. \end{aligned}$$

We then conclude that if $f = \chi_i$, then $\operatorname{obj}(f) = 1 - \epsilon$; and if $f = \chi_S$,

$$\operatorname{obj}(f) = \begin{cases} \frac{1}{2} + \frac{1}{2}(1-2\epsilon)^{|S|}, & \text{if } |S| \text{ is odd;} \\ \frac{1}{2}, & \text{if } |S| \text{ is even.} \end{cases}$$

So f has $1/\epsilon^2$ Fourier coefficients of ϵ, leading to

$$\operatorname{obj}(f) \leq \frac{1}{2} + \frac{1}{2}\frac{1}{\epsilon^2}\epsilon^3 = \frac{1}{2} + \frac{\epsilon}{2}.$$

Although this is nice and is what we want, but to do the full reduction from label cover to 3LIN, we will need a lot more work.

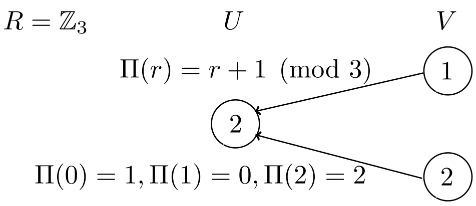
7.4 Unique Games

Thankfully, to prove Theorem 7.1.1, we can instead show the reduction from unique games to 3LIN.[1]

Problem 7.4.1 (Unique game). Given a d-regular bipartite graph $\mathcal{G} = (U \sqcup V, \mathcal{E})$ with $|U| = |V| = n$, with two label sets $R \sqcup R$ (one for each U, V) such that for all $e = (u, v) \in \mathcal{E}$, we have a bijection $\Pi_e \colon [R] \to [R]$. The *label cover* problem asks for an assignment $\sigma \colon U \sqcup V \to R$ maximizes the satisfied edges.

[1] Although as we noted, this can be done directly without any conjecture.

Remark. We see that the unique game problem is a special case of label cover, where now the label sets are the same on both sides, and the projection becomes a bijection, i.e., we now have uniqueness.



The uniqueness plays an important role here: since an assignment to one vertex uniquely determines all its neighbors, which implies the following.

Claim. The $(1, 1)$-Gap unique game has a polynomial time algorithm.

Proof. Assume that $\mathcal{G}$ is connected, then pick an arbitrary vertex v, we just try all labels and propagate. If there is an assignment satisfying every edge, we can find it this way. ⊛

On the other hand, given a label cover instance, even if we are told that there is a perfect assignment, we still can't find it. But interestingly, we hypothesize the following.

Conjecture 7.4.1 (Unique games conjecture [Kho02])**.** For every $\epsilon > 0$, there exists $R = R(\epsilon)$ such that the $(1 - \epsilon, \epsilon)$-Gap unique game is NP-hard with R.

While Theorem 6.4.1 states that the $(1, \epsilon)$-Gap label cover is NP-hard, unique game conjecture suggests that the only difference between unique game and label cover is at the $(1, 1)$-Gap version, i.e., we can solve the exact unique game, but this is the only thing we can do additionally compared to label cover.

Remark (Optimal hardness)**.** Assuming the unique game conjecture, for all $\epsilon > 0$, it is NP-hard to

(a) $(2 - \epsilon)$-approximate vertex cover and feedback vertex set;

(b) c-approximate multicut for all $c > 1$;

(c) $(\alpha_{\mathrm{GW}} + \epsilon)$-approximate max cut.

And the above are all optimal since we have seen the corresponding approximation algorithms before.

Additionally, if the unique game conjecture is true, then the Lasserre hierarchy is the best algorithms for all CSPs.

Lecture 24: From Unique Games to 3LIN

7.4.1 Unique Games to 3LIN 28 Nov. 10:30

We now show Theorem 7.1.1, i.e., the $(1-\epsilon_1, 1/2+\epsilon_1)$-Gap 3LIN is NP-hard for every $\epsilon_1 > 0$ by reduction from unique game. Given a unique game instance $\mathcal{G} = (U \sqcup V, \mathcal{E})$ and bijection $\{\Pi_e\}_{e \in \mathcal{E}}$ with label set R, we construct a 3LIN instance as usual (parametrized by ϵ):

- Variable: $(U \sqcup V) \times \{\pm 1\}^R$
- Assignment: $C_v = \{v\} \times \{\pm 1\}^R$. And given assignment, $f_v \colon \{\pm 1\}^R \to \{\pm 1\}$ be assignment restricted to C_v.
- Equation:[2] We sample an equation such that $w(\text{eq.}) = \Pr(\text{eq. is sampled})$ by the following.

[2] Again, described in probability language.

- Sample $(u, v) \in \mathcal{E}$, and $x \in \{\pm 1\}^R, y \in \{\pm 1\}^R, b \in \{\pm 1\}$.
- For all $i \in [R]$,

$$z_i = \begin{cases} x_{\Pi_e(i)} y_i b, & \text{w.p. } 1-\epsilon; \\ -x_{\Pi_e(i)} y_i b, & \text{w.p. } \epsilon. \end{cases}$$

- Output $f_u(x) f_v(y) f_v(z) = b$.

Remark. We can instead let $f_u(x) f_v(y) f_u(z) = b$ (with some index-changes) since in this case, u and v are symmetric. But if we do the reduction from label cover, it's important that we assign z to the larger side (the right-side), i.e., v.

To show the completeness, there exists $\sigma\colon U \sqcup V \to R$ that satisfies $(1-\epsilon_0)$ fraction of $\mathcal{E}$. Then the solution for 3LIN is $(v, x) \leftarrow x_{\sigma(v)}$, i.e., $f_v = \chi_{\{\sigma(v)\}}$, or by using the notion of dictator, $f_v g \chi_{\sigma(v)}$. Given that $e = (u, v)$ is sampled and e is satisfied. Then $\sigma(u) = \Pi_e(\sigma(v))$, so the equation is satisfied if $f_u(x) f_v(y) f_v(z) = b$, i.e.,

$$x_{\sigma(u)} y_{\sigma(v)} z_{\sigma(v)} = b \Leftrightarrow x_{\Pi_e(\sigma(v))} y_{\sigma(v)} z_{\sigma(v)} = b,$$

which implies the equation is satisfied with probability at least $1-\epsilon$. Finally, since we assume that there is $(1-\epsilon_0)$ fraction of $\mathcal{E}$ is satisfied, we know that the total fraction of satisfied equations is at least $(1-\epsilon_0)(1-\epsilon)$, which completes the completeness.

To show the soundness, given $\{f_v\}_{v \in U \sqcup V}$, fix $e = (u, v)$. Now, let $g \coloneqq f_v$ and

$$f(x) \coloneqq f_u(x') \text{ where } x_i = x'_{\Pi_e(i)} .$$

Note. We're enforcing f and g to use the same coordinate essentially. Furthermore, in this case, for all $i \in [R]$

$$z_i = \begin{cases} x_i y_i b, & \text{w.p. } 1-\epsilon; \\ -x_i y_i b, & \text{w.p. } \epsilon, \end{cases}$$

i.e., we permute the coordinate back in terms of z, which simplifies the analysis.

Given e,

$$\begin{aligned} \Pr_{x', y, z, b}(f_u(x') f_v(y) f_v(z) = b) &= \Pr_{x, y, z, b}(f(x) g(y) g(z) = b) \\ &= \frac{1}{2} + \frac{1}{2} \mathbb{E}_{x,y,z,b}\left[f(x) g(y) g(z) b\right] \\ &= \frac{1}{2} + \frac{1}{2} \sum_{S,T,U \subseteq [R]} \hat{f}(S) \hat{g}(T) \hat{g}(U) \mathbb{E}\left[\chi_S(x) \chi_T(y) \chi_U(z) b\right]. \end{aligned}$$

We now see that we get back $\mathbb{E}\left[\chi_S(x) \chi_T(y) \chi_U(z) b\right]$!

As previously seen. From the previous analysis, we have

$$\mathbb{E}\left[\chi_S(x) \chi_T(y) \chi_U(z) b\right] = \begin{cases} (1-2\epsilon)^{|S|}, & \text{if } S = T = U \text{ with } |S| \text{ odd}; \\ 0, & \text{otherwise.} \end{cases}$$

Hence, we have

$$\begin{aligned}
\Pr_{x',y,z,b}(f_u(x')f_v(y)f_v(z)=b) &= \frac{1}{2}+\frac{1}{2}\sum_{S \text{ odd}} \underbrace{\hat{f}(S)\hat{g}(S)^2(1-2\epsilon)^{|S|}}_{(\hat{f}(S)\hat{g}(S)(1-2\epsilon)^{2|S|})\cdot \hat{g}(S)} \\
&\leq \frac{1}{2}+\frac{1}{2}\left(\sum_{S \text{ odd}} \hat{f}(S)^2\hat{g}(S)^2(1-2\epsilon)^{2|S|}\right)^{1/2} \underbrace{\left(\sum_{S \text{ odd}} \hat{g}(S)^2\right)^{1/2}}_{\leq 1} \\
&\leq \frac{1}{2}+\frac{1}{2}\underbrace{\left(\sum_{S \text{ odd}} \hat{f}(S)^2\hat{g}(S)^2(1-2\epsilon)^{2|S|}\right)^{1/2}}_{:=\delta(e)}.
\end{aligned}$$

Now, to construct a unique game instance, for all $v \in U \sqcup V$, we sample $S \subseteq [R]$ with probability $\hat{f}(S)^2$, and then we just let $\sigma(v)$ to be a random element from S. Let $t \in \mathbb{N}$ to be undetermined. We say S is *big* if $|S|>t$, otherwise *small*. Then

$$\begin{aligned}
\delta(e) &\leq \sum_{S \text{ odd, small}} \hat{f}(S)^2\hat{g}(S)^2 + \sum_{S \text{ odd, big}} \hat{f}(S)^2\hat{g}(S)^2(1-2\epsilon)^t \\
&= \sum_{S \text{ odd, small}} \hat{f}(S)^2\hat{g}(S)^2 + (1-2\epsilon)^t \underbrace{\sum_{S \text{ odd, big}} \hat{f}(S)^2\hat{g}(S)^2}_{\leq \sum_{S,T} \hat{f}(S)^2\hat{g}(S)^2=1} \\
&\leq \sum_{S \text{ odd, small}} \hat{f}(S)^2\hat{g}(S)^2 + (1-2\epsilon)^t.
\end{aligned}$$

Formally, given a 3LIN instance, $\{f_v\}_v$ with obj $>1/2+\gamma$, we have at least $\gamma/2$ fraction of $e \in \mathcal{E}$ has $\mathrm{obj}_e \geq 1/2+\gamma/2$.

> **Note.** This is true, since otherwise, the objective value is at most
>
> $$\frac{\gamma}{2}\cdot 1+\left(1-\frac{\gamma}{2}\right)\left(\frac{1}{2}+\frac{\gamma}{2}\right) < \frac{1}{2}+\gamma,$$
>
> which is a contradiction.

Fix a good $e=(u,v)$ with $\delta(e)\geq\gamma^2$, i.e.,

$$\sum_{S \text{ odd, small}} \hat{f}(S)^2\hat{g}(S)^2 + (1-2\epsilon)^t \geq \gamma^2.$$

Recall that $g=f_v$, $f(x)=f_u(x')$ with $x_i=x'_{\Pi(i)}$, we have

$$\hat{f}(S)=\hat{f}_u(\Pi(S)) \text{ where } \Pi(S)g\left\{\Pi(i)\colon i\in S\right\},$$

so we have

$$\sum_{S \text{ odd, small}} \hat{f}_v(S)^2\hat{f}_u(\Pi(S))^2 \geq \gamma^2-(1-2\epsilon)^t.$$

Then, we have

$$\Pr(e \text{ is satisfied by the UG assignment}) \geq \frac{\gamma^2-(1-2\epsilon)^t}{t},$$

implying

$$\Pr_{\text{overall}}(\text{UG assignment satisfied}) \geq \left(\frac{\gamma}{2}\right)\left(\frac{\gamma^2-(1-2\epsilon)^t}{t}\right).$$

Now, to finish everything, we let $\epsilon_1 := \gamma > 0$, $\epsilon := \gamma/2$ and $\epsilon_0 := \gamma^5/4$. Then, we know that by letting $t=\lceil 1/\gamma^2\rceil$,

$$(1-2\epsilon)^t \leq e^{-2\epsilon t} \leq e^{-1/\gamma}.$$

So the completeness follows from the fact that if $\mathsf{OPT}_{\mathrm{UG}} \geq 1-\epsilon_0$,

$$\mathsf{OPT}_{\mathrm{3LIN}} \geq (1-\epsilon)(1-\epsilon_0) \geq 1-\epsilon-\epsilon_0 \geq 1-\gamma = 1-\epsilon_0.$$

Also, the soundness follows from that fact that if $\mathsf{OPT}_{\mathrm{3LIN}} \geq 1/2+\gamma = 1/2+\epsilon_0$, $\mathsf{OPT}_{\mathrm{UG}} \geq \gamma = \epsilon_0$.

Lecture 25: Hardness of Max-Cut

7.5 Hardness of Max-Cut

30 Nov. 10:30

In this section, we will see that if we assume the unique game conjecture, then Theorem 5.1.2 is actually the best we can do, i.e., we have the following.

Theorem 7.5.1 ([Kho+04]). Assuming the unique game conjecture, for every $\epsilon > 0$, it's NP-hard to approximate max cut within a factor of $\alpha_{\mathrm{GW}} + \epsilon$.

As previously seen. Recall that the Goemans-Williamson constant α_{GW} is defined as

$$\alpha_{\mathrm{GW}} \coloneqq \max_{a\in[-1,1]} \frac{\arccos(a)/\pi}{(1-a)/2} \approx 0.878,$$

which is the approximation ratio achieved by Algorithm 5.1.

Just like 3LIN, we design a dictator-ship test, namely to create an instance (a graph) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ for $\mathcal{V} \coloneqq \{\pm 1\}^R$ such that

- *dictator cuts* (for some $i \in [R]$, $S = \{x \in \{\pm 1\}^R : x_i = 1\}$) get good value (cut many edges);
- cuts *far from dictator* get small value (don't cut many edges).

Such a test is done is again done via noisy hypercube construction.

7.5.1 Noisy Hypercube

To start with, let $\rho \in [-1, +1]$, let $\mathcal{G}_\rho = (\mathcal{V}, \mathcal{E})$ with edge-weight such that the total weight sums up to 1. This allows us to sample an edge by first sample $x \in \{\pm 1\}^R$ uniformly, and for all $i \in [R]$, let

$$y_i = \begin{cases} x_i, & \text{w.p. } (1+\rho)/2; \\ -x_i, & \text{w.p. } (1-\rho)/2, \end{cases}$$

and output $(x, y) \in \mathcal{E}$.

Notation. For any $(x, y) \in \mathcal{E}$ in $\mathcal{G}_\rho$, we denote $x \underset{\rho}{\sim} y$.

We see that for all $i \in [R]$, $\mathbb{E}_{x \underset{\rho}{\sim} y}[x_i y_i] = 1 \cdot (1+\rho)/2 - 1 \cdot (1-\rho)/2 = \rho$.

Note. At the end, we'll try $\rho < 0$.

Let $T_\rho \in \mathbb{R}^{2^R \times 2^R}$ be the normalized adjacency matrix of $\mathcal{G}_\rho$ such that

$$T_\rho(x, y) = \Pr(y \text{ is sampled} \mid x \text{ is sampled}) = \Pr((x, y) \text{ is sampled}) \cdot 2^R.$$

We see that the sum of entries is 2^R, and rows and columns sum up to 1. Given a cut (S, T), consider $f\colon \mathcal{V} \to \{\pm 1\}$ such that

$$f(x) = \begin{cases} 1, & \text{if } x \in S; \\ -1, & \text{if } x \in T. \end{cases}$$

Claim. Given (S, T), the total weight of edges cut by (S, T) in $\mathcal{G}_\rho$ is equal to

$$\frac{1}{2}(1 - \langle f, T_\rho f \rangle).$$

Proof. The total weigh of cut edges is equal to

$$\mathbb{E}_{x \underset{\rho}{\sim} y} \left[\frac{1}{2}(1 - f(x)f(y)) \right] = \sum_{(x,y) \in \mathcal{E}} w(x, y) \cdot \frac{1}{2}(1 - f(x)f(y)).$$

Then, since

$$\mathbb{E}_{x \underset{\rho}{\sim} y} [f(x)f(y)] = \sum_{x \in \mathcal{V}} \sum_{y \in \mathcal{V}} \frac{1}{2^R} \Pr(y \mid x) f(x) f(y) = \frac{1}{2^R} \sum_{x \in \mathcal{V}} f(x) \underbrace{\sum_{y \in \mathcal{V}} \Pr(y \mid x) f(y)}_{(T_\rho f)(x) = \mathbb{E}_{x \underset{\rho}{\sim} y}[f(y)]} = \langle f, T_\rho f \rangle,$$

so we're done. ⊛

Definition 7.5.1 (Stability). The *stability* of f given ρ is defined as

$$\mathrm{Stab}_\rho(f) \coloneqq \langle f, T_\rho f \rangle = \mathbb{E}_{x \underset{\rho}{\sim} y} [f(x)f(y)].$$

Remark. If f is ± 1-valued, then $\mathrm{Stab}_\rho(f) = 2 \Pr_{x \underset{\rho}{\sim} y}(f(x) = f(y)) - 1$. So if $f(x) \approx f(y)$ for y being the neighbor of x (i.e., $x \underset{\rho}{\sim} y$), this quantity is stable and close to 1.

To study $\mathrm{Stab}_\rho(f)$, consider $S \subseteq [R]$ and χ_S, we have

$$\mathrm{Stab}_\rho(\chi_S) = \mathbb{E}_{x \underset{\rho}{\sim} y} \left[x^S y^S \right] = \prod_{i \in S} \mathbb{E}[x_i y_i] = \rho^{|S|}$$

since $\chi_S(x) = x^S = \prod_{i \in S} x_i$. Now, since

$$f = \sum_{S \subseteq [R]} \hat{f}(S) \chi_S,$$

we have

$$\langle f, T_\rho f \rangle = \sum_{S,T} \hat{f}(S) \hat{f}(T) \langle \chi_S, T_\rho \chi_T \rangle = \sum_{S \subseteq [R]} \hat{f}(S)^2 \cdot \rho^{|S|}$$

since $T_\rho \chi_T = \rho^{|T|} \cdot \chi_T$. So if $f = \chi_i$ for some $i \in [R]$, the cut value is equal to

$$\frac{1}{2} - \frac{1}{2} \mathrm{Stab}_\rho(f) = \frac{1}{2} - \frac{1}{2} \rho$$

for $\rho < 0$, so a dictator cuts indeed get good value.

Problem. How we define *far from dictators* and prove they have small values?

7.5.2 Majority is The Stablest Theorem

Consider $f \colon \{\pm 1\}^R \to \{\pm 1\}$ as voting rules:

- R people;
- $x \in \{\pm 1\}^R$ corresponds to a voting outcome;
- Society goes with $f(x)$.

Example (Majority). We can take $f(x)$ to be the majority vote, i.e.,

$$f(x) = \mathrm{Maj}(x) \coloneqq \mathrm{sgn}(x_1 + \cdots + x_R).$$

Example (Dictator). We can also take $f(x)$ to be following a dictator i, i.e.,

$$f(x) = \mathrm{Dictator}_i(x) \coloneqq x_i.$$

Definition 7.5.2 (Influence). Given $f\colon \{\pm 1\}^R \to \{\pm 1\}$, for an $i \in [R]$, the *influence* $\mathrm{Inf}_i(f)$ of i is defined as

$$\mathrm{Inf}_i(f) \coloneqq \Pr_x(f(x) \neq f(x \oplus e_i)) = \sum_{S \ni i} \hat{f}(S)^2.$$

But one problem of Definition 7.5.2 is that the total influence is not the same for all f, since

$$\mathrm{I}(f) \coloneqq \sum_{i \in [R]} \mathrm{Inf}_i(f) = \sum_{S \subseteq [R]} \hat{f}(S)^2 |S|.$$

Note. Indeed, the total influence can vary a lot, since $\mathrm{I}(\chi_i) = 1$ while $\mathrm{I}(\chi_{[R]}) = |R|$.

To fix this issue, we define the so-called

Definition 7.5.3 (δ-noisy influence). Given $f\colon \{\pm 1\}^R \to \{\pm 1\}$ and $\delta \in (0, 1)$, for an $i \in [R]$, the *δ-noisy influence* $\mathrm{Inf}_i^\delta(f)$ of i is defined as

$$\mathrm{Inf}_i^\delta(f) \coloneqq \sum_{S \ni i} \hat{f}(S)^2 (1-\delta)^{|S|-1}.$$

In this case, the total δ-noisy influence $\mathrm{I}^\delta(f)$ is given by

$$\mathrm{I}^\delta(f) = \sum_{S \subseteq [R]} \hat{f}(S)^2 |S(1-\delta)|^{|S|-1}.$$

Claim. We have $\mathrm{I}^\delta(f) \leq 1/\delta \|f\|_2^2$.

Proof. We simply maximize $|S|(1-\delta)^{|S|-1}$ for every S, where we simply have $1/\delta$. ⊛

Now, consider the majority vote $\mathrm{Maj}\colon \{\pm 1\}^R \to [\pm 1]$, we have

$$\mathrm{Inf}_i(\mathrm{Maj}) = \Theta(1/\sqrt{R}), \quad \mathrm{I}(\mathrm{Maj}) = \Theta(\sqrt{R});$$

while

$$\mathrm{Inf}_i^\delta(\mathrm{Maj}) = \Theta(1), \quad \mathrm{I}^\delta(\mathrm{Maj}) = \Theta(1/R).$$

Now, we say that i is *influential* if $\mathrm{Inf}_i^\delta(f) \geq \Omega(1)$.

Claim. As $R \to \infty$, $\mathrm{Stab}_\rho(\mathrm{Maj}) \to 2/\pi \cdot \arcsin \rho$.

Proof. Firstly, let

$$X = \frac{x_1 + \cdots + x_R}{\sqrt{R}}, \quad Y = \frac{y_1 + \cdots + y_R}{\sqrt{R}},$$

we have $\mathbb{E}[X] = \mathbb{E}[Y] = 0$, $\mathbb{E}[X^2] = \mathbb{E}[Y^2] = 1$ with $\mathbb{E}[XY] = \rho$, we have

$$\begin{aligned}\mathrm{Stab}_\rho(\mathrm{Maj}) &= \mathbb{E}_{x \underset{\rho}{\sim} y}\left[\mathrm{sgn}(x_1 + \cdots + x_R)\cdot \mathrm{sgn}(y_1 + \cdots + y_R)\right] \\ &= \mathbb{E}_{X,Y}\left[\mathrm{sgn}(X)\,\mathrm{sgn}(Y)\right] \\ &\approx \mathbb{E}_{g,h}\left[\mathrm{sgn}(g)\,\mathrm{sgn}(h)\right]\end{aligned}$$

where $g, h \sim \mathcal{N}(0,1)$ are ρ-correlated by 2-dimensional central limit theorem. Notice that we interpret the term ρ-correlated as $g = \langle a, t\rangle = t_1$, $h = \langle b, t\rangle$ where we sample $t := (t_1, t_2) \sim \mathcal{N}(0,1)^2$.[a] In this case, we have the same geometric interpretation as in Lemma 5.1.4, we have

$$\Pr(\mathrm{sgn}(g) = \mathrm{sgn}(h)) = \frac{\pi - \arccos\rho}{\pi}.$$

⊛

[a] We implicitly rotate t_1 to the x-axis from $g = \langle a, t\rangle = t_1$.

Finally, we need the following theorem, which is introduced in [Kho+04], and confirmed in [MOO05] later.

Theorem 7.5.2 (Majority is the stablest [MOO05; Kho+04]). Given $0 < \rho < 1$ and $\epsilon > 0$, and $f\colon \{\pm 1\}^n \to [-1, 1]$ with $\mathbb{E}[f] = 0$. Suppose $\mathrm{Inf}_i^{1/\log(1/\epsilon)}(f) \leq \epsilon$, then

$$\mathrm{Stab}_\rho(f) \leq \left(\frac{2}{\pi}\right)\arcsin\rho + O\left(\frac{\log\log 1/\epsilon}{\log 1/\epsilon}\right).$$

Lecture 26: A Final Reduction from Unique Games

7 Dec. 10:30

Proof sketch. From $\mathrm{Stab}_\rho[f] = \mathbb{E}_{x \underset{\rho}{\sim} y}[f(x)f(y)]$, we want to say that this is equal to $\mathbb{E}_{g \underset{\rho}{\sim} h}[f(h)f(h)]$ where $g \underset{\rho}{\sim} h$ means for all $i \in [R]$, $g_i \sim \mathcal{N}(0,1)$, $h_i = \rho \cdot g + \sqrt{1-\rho^2}\mathcal{N}(0,1)$. In this case, though $f\colon \{0,1\}^{[R]} \to \{0,1\}$ doesn't make sense, but if we look at the Fourier decomposition, we have

$$f(x) = \sum_{S \subseteq [R]} \hat{f}(S) x^S \Rightarrow f(g) = \sum_{S \subseteq [R]} \hat{f}(S) g^S.$$

Remark (Invariance principle). If f is low-influence, then $(f(x), f(y))$ is similarly distributed as $(f(g), f(h))$.

Now, consider $f\colon \mathbb{R}^R \to \{\pm 1\}$ such that $\mathbb{E}[f] = 0$. With the mean condition, f is essentially deciding a set with the measure being a half. Furthermore, for $g \underset{\rho}{\sim} h$, we want if g is in the set, h will be in the set as well. This is done by minimize the boundary of the set f is deciding, which turns out to be a half-space. This corresponds to the majority vote directly, so we're done. ■

Corollary 7.5.1. Given $\rho < 0$ and $\epsilon > 0$, and $f\colon \{\pm 1\}^n \to [-1, 1]$ with $\mathbb{E}[f] = 0$. Suppose $\mathrm{Inf}_i^{1/\log(1/\epsilon)}(f) \leq \epsilon$, then

$$\mathrm{Stab}_\rho(f) \geq \left(\frac{2}{\pi}\right)\arcsin\rho + O\left(\frac{\log\log 1/\epsilon}{\log 1/\epsilon}\right).$$

7.5.3 Reduction from Unique Game

Finally, we can now prove Theorem 7.5.1. Let $\rho \approx -0.7$, such that

$$c = \frac{1}{2} - \frac{1}{2}\rho \approx 0.85, \quad s = \frac{1}{2} - \frac{1}{\pi}\arcsin\rho \approx 0.75,$$

and $c/s = \alpha_{\mathrm{GW}}$. Fix $\epsilon_1 > 0$, choose ϵ_0 and give reduction from $(1-\epsilon_0, \epsilon_0)$-Gap unique game to $(c-\epsilon_1, s+\epsilon_1)$-Gap max cut. Given a unique game instance $\mathcal{G} = (U \sqcup V, \mathcal{E})$ which is regular, $\{\Pi_e\}_{e\in\mathcal{E}}$, $[R]$, our max cut instance is designed as

- Vertex: $U \times \{\pm 1\}^R$.
- Edges:
  - sample $v \in V$
  - sample $u_1, u_2 \sim N(v)$[3]
  - sample $x \in \{\pm 1\}^R$, and $x \underset{\rho}{\sim} y$[4]
  - output $\big((u_1, \Pi_1(x)), (u_2, \Pi_2(y))\big)$ where $\big(\Pi_1(x)\big)_{\Pi_1(i)} = x_i$.

Completeness

To show completeness, consider a unique game instance $\mathcal{G} = (U \sqcup V, \mathcal{E})$ with a labeling $\sigma\colon U \sqcup V \to [R]$ which satisfies $1-\epsilon_0$ fraction of $\mathcal{E}$. Let the cut indicated by $f_u(x) = x_{\sigma(u)}$ for all $u \in U$ be

$$S_u \coloneqq \big\{(u,x)\colon x_{\sigma(u)} = 1\big\}.$$

By union bound, with probability $\geq 1 - 2\epsilon_0$, we have

$$\Pr(f_{u_1}(\Pi_1(x)) \neq f_{u_2}(\Pi_2(y))) = \Pr(\big(\Pi_1(x)\big)_{\sigma(u_1)} \neq \big(\Pi_2(y)\big)_{\sigma(u_2)}) = \Pr(X_{\sigma(v) \neq y_{\sigma(v)}}) = \frac{1-\rho}{2},$$

where $\Pi_1(\sigma(u)) = \sigma(u_1)$ and $\Pi_2(\sigma(u)) = \sigma(u_2)$. This means that S cuts at least $(1-2\epsilon_0) \times (1-\rho)/2$ fraction of edges.

Soundness

To prove soundness, suppose $\{f_u\}_{u\in U}$ such that $f_u\colon \{\pm 1\}^R \to \{\pm 1\}$ is the assignments to the max cut vertices with cut value $\geq 1/2 - \arcsin \rho/\pi + \epsilon_1$. Our goal is to get a labeling σ for the unique game with value at least ϵ_0. Firstly, we have the following.

Claim. Given the sampling procedure,

$$\mathbb{E}_{v,x,y}\left[f_v(x) f_v(y)\right] \leq \left(\frac{2}{\pi}\right) \arcsin \rho - 2\epsilon_1.$$

Proof. For all $v \in V$, $f_v\colon \{\pm 1\}^R \to [-1,1]$ such that $f_v(x) \coloneqq \mathbb{E}_{u\sim N(v)}\left[f_u(\Pi_{uv}(x))\right]$. Then, if we sample $v \in V$ and $x \underset{\rho}{\sim} y$, we have

$$\begin{aligned}
\mathbb{E}_{v,u_1,u_2,x,y}\left[f_{u_1}(\Pi_1(x)) f_{u_2}(\Pi_2(y))\right] &= \mathbb{E}_{v,x,y}\left[\mathbb{E}_{u_1,u_2}\left[f_{u_1}(\Pi_1(x)) f_{u_2}(\Pi_2(y)) \mid v,x,y\right]\right] \\
&= \mathbb{E}_{v,x,y}\left[\mathbb{E}_{u_1}\left[f_{u_1}(\Pi_1(x)) \mid v,x\right] \cdot \mathbb{E}_{u_2}\left[f_{u_2}(\Pi_2(y)) \mid v,y\right]\right] \\
&= \mathbb{E}_{v,x,y}\left[f_v(x) f_v(y)\right],
\end{aligned}$$

so with the fact that[a]

$$\mathbb{E}_{v,u_1,u_2,x,y}\left[f_{u_1}(\Pi_1(x)) f_{u_2}(\Pi_2(y))\right] \leq \left(\frac{2}{\pi}\right) \arcsin \rho - 2\epsilon_1,$$

we're done. ⊛

[a]This follows from the sampling procedure.

[3]$N(v)$ is the neighborhood of v, so $e_1 = (u_1, v), e_2 = (u_2, v) \in \mathcal{E}$.

[4]x is a vertex from the hypercube of u_1, and y is a vertex from the hypercube of u_2. This is the dictator-ship test in subsection 7.5.1.

By Markov inequality, at least ϵ_1 fraction of v satisfies

$$\mathbb{E}_{x,y}\left[f_v(x)f_v(y)\right] \leq \left(\frac{2}{\pi}\right)\arcsin\rho - \epsilon_1,$$

and we say that such v is *good*. From Theorem 7.5.2, for any good v, there exists some $\epsilon > 0$ and $i \in [R]$ such that $\mathrm{Inf}_i^{\delta}(f_v) \geq \epsilon$ with some large enough $\delta = \delta(\epsilon, \rho) > 0$. Naturally, we label u as $\sigma(u) \coloneqq i$.

Note the following.

Remark (Fact). Given the $\epsilon > 0$ found above, $\epsilon \leq \mathbb{E}_u\left[\mathrm{Inf}^i_{\Pi_{uv}(i)}(f_u)\right]$.

Proof. We have

$$\begin{aligned}
\epsilon \leq \mathrm{Inf}_i^{\delta}(f_v) &= \sum_{S\ni i}\hat{f}_v(S)^2(1-\delta)^{|S|-1} && \text{from } \hat{f}(S) = \mathbb{E}_u\left[\hat{f}_u(\Pi_{uv}(S))\right]\\
&= \sum_{S\ni i}(1-\delta)^{|S|-1}\mathbb{E}_u\left[\hat{f}_u(\Pi_{uv}(S))\right]^2\\
&\leq \mathbb{E}_u\left[\sum_{S\ni i}(1-\delta)^{|S|-1}\hat{f}_u(\Pi_{uv}(S))^2\right]\\
&= \mathbb{E}_u\left[\mathrm{Inf}^{\delta}_{\Pi_{uv}(i)}(f_u)\right].
\end{aligned}$$

⊛

This fact implies that at least $\epsilon/2$ fraction of the neighbors u of v we have $\mathrm{Inf}^{\delta}_{\Pi_{uv}(i)}(f_u) \geq \epsilon/2$, and similarly, we call such u *good*. To label any $u \in U$, we consider the set of candidates $\{j \in [R] \colon \mathrm{Inf}^{\delta}_j(f_u) \geq \epsilon/2\}$, hence for all good u, $\Pi_{uv}(i)$ is one of the candidate. Observe that since

$$\sum_{i\in[R]}\mathrm{Inf}^{\delta}_i(f_u) = \mathrm{I}^{\delta}(f_u) \leq \delta,$$

so the number of candidate of u is at most $2\delta/\epsilon$. We now simply label u uniformly at random from the set of candidates. In this case, for any good $v \in V$ and $u \in U$, if u we choose is exactly $\Pi_{uv}^{-1}(\sigma(v))$, then this edge uv is satisfied. From a simple calculation, we know that at least

$$\epsilon_0 \coloneqq \frac{\epsilon_1}{2} \times \frac{\epsilon}{2} \times \frac{2\delta}{\epsilon} = \frac{\epsilon_1\delta}{2}$$

fraction of the edges are satisfied, completing the proof.

Remark. Without assuming unique game conjecture, it's only known to be NP-hard to approximate max cut with ratio better than 0.92. And with Theorem 7.5.1, if unique game conjecture, α_{GW} is optimal.

Appendix

Appendix A

Review

A.1 Boolean Satisfaction Problem

Here, we give a quick review toward the MAX-3SAT problem.

Definition A.1.1 (Conjunctive normal form). A *conjunctive normal form* (CNF) formula is a conjunction φ of one or more boolean clauses on $x_1, x_2, \ldots, x_n$ with boolean valued $\{0, 1\}$. Explicitly, φ is in CNF if

$$\varphi(x_1, x_2, \ldots, x_n) = \text{clause}_1 \wedge \text{clause}_2 \wedge \text{clause}_3 \wedge \cdots \wedge \text{clause}_k$$

where each clause is an or of literals, with a literal being some x_i or its negation $\neg x_i$.

Note (Disjunctive normal form). For every conjunctive normal form, there is an equivalent way to write it in the so-called *disjunctive normal form*.

Definition A.1.2 (k-CNF). A *k-CNF formula* is a CNF formula in which each clause has exactly k literals from distinct variables.

Example (3-CNF). A 3-CNF formula can be like

$$\varphi = (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_3 \vee x_4 \vee x_5) \wedge (\neg x_1 \vee \neg x_5 \vee \neg x_6).$$

Now, the boolean satisfability problem asks the following question: given a k-CNF formula φ, does an assignment exist such that φ is evaluated as true? Formally, we have Problem A.1.1.

Problem A.1.1 (k-SAT). Given a k-CNF formula φ, the *k-SAT* problem asks whether φ is satisfiable.

Instead of looking at a general k, we consider a simple but also hard enough case when $k = 3$. Specifically, we ask the following question.

Problem A.1.2 (MAX-3SAT). Given a 3-CNF formula φ and ℓ, the *MAX-3SAT* problem asks is there an assignment of variables such that it satisfies at least ℓ clauses?

Remark. We often call MAX-3SAT as 3SAT for brevity.

A.1.1 Random MAX-3SAT

A surprising result is that by randomly assigning x_i, we achieve the best we can do in expectation. Algorithmically, we have the following.

Algorithm A.1: Deterministic MAX-3SAT

Data: A 3-CNF $\varphi(x_1, \ldots, x_n)$
Result: A 7/8-approximation assignment $\{x_i\}_{i=1}^n$ (in expectation)

for $i = 1, \ldots, n$ **do**
  $x_i \leftarrow$ `uniform`$(\{0, 1\})$ `// random assignments`
return $\{x_i\}_{i=1}^n$

Lemma A.1.1. For all 3-CNF φ, there exists an assignment that satisfies at least 7/8 of clauses.

Proof. Each clause is satisfied by all but exactly 1 assignment, the one where all the literals in the clause evaluate to false. Imagine we have a uniformly random assignment of variables, and let X_i be a random variable such that

$$X_i = \begin{cases} 1, & \text{if } i^{st} \text{ clause is satisfied;} \\ 0, & \text{otherwise.} \end{cases}$$

Since each clause has 8 different possibilities (2^3), and there is only one situation where the clause is not satisfied, each clause has a probability of 7/8 of being satisfied. Therefore:

$$\Pr(X_i = 1) = \frac{7}{8} = \mathbb{E}[X_i].$$

Let X be a random variable that corresponds to the number of satisfied clauses, i.e., $X = \sum_{i=1}^n X_i$. By the linearity of expectation, we have

$$\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}(X_i) = \frac{7}{8} \cdot m \geq \frac{7}{8}\,\mathsf{OPT}$$

This shows that if we had an algorithm that randomly picks assignments of variables and checks to see how many clauses are satisfied, this would be a randomized algorithm that achieves $\alpha = \frac{7}{8}$. If we were then to repeat the algorithm a polynomial number of times, we could show that there is a good chance to find such an assignment using Markov's inequality. ■

Bibliography

[Aro+98] Sanjeev Arora et al. "Proof Verification and the Hardness of Approximation Problems". In: *J. ACM* 45.3 (May 1998), pp. 501–555. ISSN: 0004-5411. DOI: 10.1145/278298.278306. URL: https://doi.org/10.1145/278298.278306.

[BLR90] M. Blum, M. Luby, and R. Rubinfeld. "Self-Testing/Correcting with Applications to Numerical Problems". In: *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*. STOC '90. Baltimore, Maryland, USA: Association for Computing Machinery, 1990, pp. 73–83. ISBN: 0897913612. DOI: 10.1145/100216.100225. URL: https://doi.org/10.1145/100216.100225.

[BS14] Boaz Barak and David Steurer. *Sum-of-squares proofs and the quest toward optimal algorithms*. 2014. DOI: 10.48550/ARXIV.1404.5236. URL: https://arxiv.org/abs/1404.5236.

[Chr76] Nicos Christofides. "Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem". In: *Oper. Res. Forum* 3 (1976).

[Coh+22] Vincent Cohen-Addad et al. *Breaching the 2 LMP Approximation Barrier for Facility Location with Applications to k-Median*. 2022. DOI: 10.48550/ARXIV.2207.05150. URL: https://arxiv.org/abs/2207.05150.

[Coo71] Stephen A. Cook. "The Complexity of Theorem-Proving Procedures". In: STOC '71. Shaker Heights, Ohio, USA: Association for Computing Machinery, 1971, pp. 151–158. ISBN: 9781450374644. DOI: 10.1145/800157.805047. URL: https://doi.org/10.1145/800157.805047.

[Fei+91] Uriel Feige et al. "Approximating clique is almost NP-complete". In: *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science* (1991), pp. 2–12.

[Fei+96] Uriel Feige et al. "Interactive Proofs and the Hardness of Approximating Cliques". In: *J. ACM* 43.2 (Mar. 1996), pp. 268–292. ISSN: 0004-5411. DOI: 10.1145/226643.226652. URL: https://doi.org/10.1145/226643.226652.

[GK99] Sudipto Guha and Samir Khuller. "Greedy Strikes Back: Improved Facility Location Algorithms". In: *Journal of Algorithms* 31.1 (1999), pp. 228–248. ISSN: 0196-6774. DOI: https://doi.org/10.1006/jagm.1998.0993. URL: https://www.sciencedirect.com/science/article/pii/S0196677498909932.

[GW95] Michel X Goemans and David P Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM (JACM)* 42.6 (1995), pp. 1115–1145.

[Hås97] Johan Håstad. "Some Optimal Inapproximability Results". In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*. STOC '97. El Paso, Texas, USA: Association for Computing Machinery, 1997, pp. 1–10. ISBN: 0897918886. DOI: 10.1145/258533.258536. URL: https://doi.org/10.1145/258533.258536.

[JMS02] Kamal Jain, Mohammad Mahdian, and Amin Saberi. "A New Greedy Approach for Facility Location Problems". In: STOC '02. Montreal, Quebec, Canada: Association for Computing Machinery, 2002, pp. 731–740. ISBN: 1581134959. DOI: 10.1145/509907.510012. URL: https://doi.org/10.1145/509907.510012.

[JV01] Kamal Jain and Vijay V. Vazirani. "Approximation Algorithms for Metric Facility Location and K-Median Problems Using the Primal-Dual Schema and Lagrangian Relaxation". In: *J. ACM* 48.2 (2001), pp. 274–296. ISSN: 0004-5411. DOI: 10.1145/375827.375845. URL: https://doi.org/10.1145/375827.375845.

[Kar72] Richard M Karp. "Reducibility among combinatorial problems". In: *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[Kar93] David R Karger. "Global min-cuts in RNC, and other ramifications of a simple min-out algorithm". In: *SODA '93*. 1993.

[Kho+04] S. Khot et al. "Optimal inapproximability results for MAX-CUT and other 2-variable CSPs?" In: *45th Annual IEEE Symposium on Foundations of Computer Science*. 2004, pp. 146–154. DOI: 10.1109/FOCS.2004.49.

[Kho02] Subhash Khot. "On the Power of Unique 2-Prover 1-Round Games". In: *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*. STOC '02. Montreal, Quebec, Canada: Association for Computing Machinery, 2002, pp. 767–775. ISBN: 1581134959. DOI: 10.1145/509907.510017. URL: https://doi.org/10.1145/509907.510017.

[KKG21] Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. "A (Slightly) Improved Approximation Algorithm for Metric TSP". In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2021. Virtual, Italy: Association for Computing Machinery, 2021, pp. 32–45. ISBN: 9781450380539. DOI: 10.1145/3406325.3451009. URL: https://doi.org/10.1145/3406325.3451009.

[Li13] Shi Li. "A 1.488 approximation algorithm for the uncapacitated facility location problem". In: *Information and Computation* 222 (2013). 38th International Colloquium on Automata, Languages and Programming (ICALP 2011), pp. 45–58. ISSN: 0890-5401. DOI: https://doi.org/10.1016/j.ic.2012.01.007. URL: https://www.sciencedirect.com/science/article/pii/S0890540112001459.

[MOO05] Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. *Noise stability of functions with low influences: invariance and optimality*. 2005. DOI: 10.48550/ARXIV.MATH/0503503. URL: https://arxiv.org/abs/math/0503503.

[ODo21] Ryan O'Donnell. *Analysis of Boolean Functions*. 2021. DOI: 10.48550/ARXIV.2105.10386. URL: https://arxiv.org/abs/2105.10386.

[Rot13] Thomas Rothvoß. "The lasserre hierarchy in approximation algorithms". In: (2013). URL: https://sites.math.washington.edu//~rothvoss/lecturenotes/lasserresurvey.pdf.

[Ser78] Anatoliy I Serdyukov. "On some extremal walks in graphs". In: (1978), pp. 76–79.

[Vaz02] V.V. Vazirani. *Approximation Algorithms*. Springer Berlin Heidelberg, 2002. ISBN: 9783540653677. URL: https://books.google.com/books?id=EILqAmzKgYIC.

[WS11] D.P. Williamson and D.B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. ISBN: 9781139498173. URL: https://books.google.com/books?id=Cc%5C_Fdqf3bBgC.