

# Discrete Optimization

**Instructor:** Yuan Zhou

**Notes Taker:** Zejin Lin

TSINGHUA UNIVERSITY.

linzj23@mails.tsinghua.edu.cn

[lzmjmaths.github.io](https://lzmjmaths.github.io)

June 6, 2025

## Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Models of Computation: Turing Machines . . . . .	4
1.2	Models of Computation: word RAM . . . . .	5
1.3	Polynomial Running Time . . . . .	5
1.4	Notation . . . . .	5
1.5	Tentative Syllabus . . . . .	6
<b>2</b>	<b>Greedy Algorithms</b>	<b>6</b>
2.1	Interval Scheduling . . . . .	6
2.2	Interval Partitioning . . . . .	7
2.3	Single-Source Shortest Path . . . . .	8
2.4	Minimum Spanning Tree . . . . .	10

2.5	Minimum Arborescence . . . . .	13
<b>3</b>	<b>Dynamic Programming</b>	<b>15</b>
3.1	Weighted Interval Scheduling . . . . .	15
3.2	Segmented Least Square . . . . .	16
3.3	Knapsack Problem . . . . .	16
3.4	RNA Secondary Structure . . . . .	19
3.5	Sequence Alignment(Edit Distance) . . . . .	19
3.6	Matrix Multiplication . . . . .	21
<b>4</b>	<b>Flow Network</b>	<b>21</b>
4.1	Definition . . . . .	21
4.2	Appllication . . . . .	27
<b>5</b>	<b>Introduction to Approximation Algorithm</b>	<b>34</b>
5.1	Set-Cover . . . . .	36
5.2	Weighted Min Set-cover and Randomized Rounding . . . . .	38
5.3	Hardness of Approximation . . . . .	46
5.4	Label-Cover Games . . . . .	52
5.5	Multicut . . . . .	58
5.6	3-Coloring . . . . .	69
5.7	Sparsest Cut . . . . .	71
5.8	Hardness of Approximating MAXE3LIN . . . . .	79
5.9	Hardness of Max-Cut . . . . .	85
	<b>Index</b>	<b>89</b>
	<b>Important Theorems</b>	<b>90</b>



# 1 Introduction

Discrete (combinatorial) optimization is a subfield of mathematical optimization that consists of finding an optimal object from a finite set of objects, where the set of feasible solution is discrete or can be reduced to a discrete set.

However, usually this feasible solution set is very large (due to combinatorial explosion) and it is computationally infeasible to go through all feasible solutions and find the one with optimal objective function value.

**Example 1.1** (Task Assignment). There are  $n$  tasks and  $n$  workers. Each task has an importance score  $a_i$  and each worker has a skill level  $b_i$ . We need to assign each task to a worker such that the sum of  $\sum_{i=1}^n a_i b_{\sigma(i)}$  is maximized.

## 1.1 Models of Computation: Turing Machines

**Definition 1.2** (A Deterministic Turing Machine(DTM)). It consists of an infinitely-long tape (memory) and a deterministic finite automata that controls the head to move along the tape and read/write symbols from/to the tape cells.

**Definition 1.3** (Complexity measure). Running time is the number of steps of Turing machine.

Memory is the number of tape cells used.

**Definition 1.4** (Caveat). No random access of memory

- Single-tape DTM requires  $\geq n^2$  steps to detect  $n$  bit palindromes.
- EASY to detect palindromes within  $c_n$  steps on a real computer.

## 1.2 Models of Computation: word RAM

**Definition 1.5.** Each memory location and input/output cell stores a  $w$ -bit integer (assume  $w \geq \log_2 \omega$ ).

Primitive Operations:

## 1.3 Polynomial Running Time

**Definition 1.6.** We say that an algorithm is **efficient** if its running time is polynomial of input size  $n$ .

**Example 1.7** (Task machine). Polynomial-time algorithm: selection sort/inserting sort/quick sort/merge sort.

Non-polynomial-time algorithm: try all possible matching and output the one with the highest score.

- Definition is relatively insensitive to model of computation.
- The poly-times algorithm that people develop have both small constants and small exponents
- Breaking through the exponential barrier is a major challenge.

## 1.4 Notation

**Definition 1.8.**  $f(n)$  is  $O(g(n))$  if there exist constants  $c > 0$  and  $n_0 \geq 1$  such that  $0 \leq f(n) \leq c \cdot g(n)$  for all  $n \geq n_0$ .

$f(n)$  is  $\Omega(g(n))$  is  $g(n) \in O(f(n))$ .

$f(n)$  is  $\Theta(g(n))$  is both  $f(n) \in O(g(n))$  and  $g(n) \in O(f(n))$ .

## 1.5 Tentative Syllabus

We will introduce three exact discrete optimization algorithms(6 weeks):

- Greedy algorithms
- Dynamic programming
- Network flows

And some approximation algorithms for intractable discrete optimization problems(9 weeks)

- Definition of approximation algorithms
  - Algorithm techniques: greedy, linear programming relaxation, semidefinite programming relaxation.
- Hardness of approximation
  - Techniques: hardness reductions, Fourier analysis of Boolean functions.
- Problems studied: Set-Cover, facility location, K-center, Multi-Cut, Max-Cut,  $\dots$

## 2 Greedy Algorithms

### 2.1 Interval Scheduling

**Example 2.1** (Interval Scheduling). Input:  $n$  jobs,  $\{(s_i, f_i)\}_{i=1}^n$ . Goal: How to choose jobs with maximized number such that each pair of intervals do not intersect.

**Greedy Framework** Consider jobs in order  $\pi(1), \pi(2), \dots, \pi(n)$ . For each  $\pi(i)$ ,  $i = 1, 2, \dots, n$ , if  $\pi(i)$  compatible with all selected jobs, then select  $\pi(i)$ .

The choice of  $\pi$ : Earliest-start-time-first, Earliest-finish-time-first, Longest-job-first, Shortest-job-first, etc.

**Theorem 2.2.** *Earliest-finish-time-first greedy returns an optimal solution.*

*Proof.* Suppose algorithm selects  $i_1, i_2, \dots, i_k$ , opt selects  $k' > k$  jobs.

Choose an optimal solution agrees with algorithm in first  $r$  jobs so that  $r$  maximized,  $j_1, j_2, \dots, j_{k'}$ .

Obviously,  $r < k$ . Then  $f_{i_{r+1}} < f_{j_{r+1}}$ . Therefore, we can replace  $i_{r+1}$  with  $j_{r+1}$  to get another optimal solution, which contradicts to the fact that  $r$  maximized.  $\square$

## 2.2 Interval Partitioning

**Example 2.3** (Interval Partitioning). Input:  $n$  lectures,  $\{(s_i, f_i)\}_{i=1}^n$ .

Goal: Position lectures into minimum number of classrooms so that in each classroom lectures are compatible.

**Greedy Framework** Lectures in order  $\pi(1), \dots, \pi(n)$ , the number of opening classrooms is zero in the beginning. For each  $\pi(i)$ ,

If  $\exists$  opening classroom  $j$  s.t. lecture  $\pi(i)$  compatible with lectures in  $j$ , then  $\pi(i) \rightarrow$  classroom  $j$ .

Else, open a new classroom for  $\pi(i)$ .

*Proof.* Introduce a concept **Depth**:  $d(t) =$  Number of lectures active at time  $t$ , and  $d = \max_t \{d(t)\}$ .

**Claim 2.2.1.**  $\text{OPT} \geq d$ .

**Lemma 2.4.**  $\text{Alg} \leq d$

*Proof.* Assume for contradiction.

At some point, Alg opens  $d + 1$  classroom.

Denote the lecture being considered by  $i$ . Then it is not compatible with other  $d$  lectures. Hence, there should be a time when  $d + 1$  lectures are active, which causes contradiction. □

□

## 2.3 Single-Source Shortest Path

**Example 2.5** (Single-Source Shortest Path(SSSP)). Input: Graph  $G = (V, E, w)$ ,  $V$  is the set of point and  $E$  is the set of edge with direction and  $w : E \rightarrow \mathbb{R}_{\geq 0}$ .

We want to find a path from  $s$  to  $t$  with minimum total cost.

**Dijkstra's Algorithm** Choose  $s$  as a source.  $d[s] = 0, d[u] =$   

$$\begin{cases} w(s, u) & \text{if } (s, u) \in E \\ +\infty & \text{otherwise} \end{cases}, S = \{s\} \text{ first. To record the path, we can use } \text{Pred}[u] \leftarrow s.$$

---

### Algorithm 1 Dijkstra's Algorithm

---

```

1: while  $S \neq V$  do
2:   Choose  $u \in \arg \min_{x \notin S} \{d[x]\}$ .
3:   Update  $S \leftarrow S \cup \{u\}$ .
4:   for each  $x \in V - S, (u, x) \in E$  do
5:      $d[x] \leftarrow \min\{d[x], d[u] + w(u, x)\}$ .
6:     if  $d[u] + w(u, x) < d[x]$  then
7:        $d[x] \leftarrow d[u] + w(u, x)$ 
8:        $\text{Pred}[x] \leftarrow u$ 
9:     end if
10:  end for
11: end while

```

---

**Theorem 2.6** (Invariant).  $\forall u \in S, d[u]$  is the shortest path distance  $s \rightsquigarrow u$



*Proof.* Induction on  $|S|$ .

For  $|S| = 1$  true.

**Induction Step:** Every time executing 2 in Algorithm 1, we need to prove  $d[u]$  is the shortest distance  $s \rightsquigarrow u$ .

If  $v = \text{Pred}[u] \in S$ , then  $d[u] = d[v] + \omega(v, u)$ .

For any path from  $s$  to  $u$ , there exists  $(\alpha, \beta) \in E$  such that  $\alpha \in S, \beta \notin S$ . Then

$$\begin{aligned} \text{length}(P) &\geq \text{length}(P[s \rightarrow \beta]) \\ &= \text{length}(P[s \rightarrow \alpha]) + \omega(\alpha, \beta) \\ &\geq d[\alpha] + \omega(\alpha, \beta) \\ &\geq d[\beta] \geq d[u] \end{aligned}$$

□

**Remark 2.7.** The straightforward implementation of Dijkstra's Algorithm is of  $O(|V|^2)$ .

If we use priority queue:  $Q$  with priority  $Q.\pi()$ . It has some methods:

- ExtractMin: Return  $\arg \min_{x \in Q} \{Q.\pi(x)\}$  and remove  $x$  from  $Q$ .
- DecreaseKey: Update  $Q.\pi(v)$  with newkey.

The time complexity is  $|V| \times \text{ExtractMin} + |E| \times \text{DecreaseKey}$

Runtime	ExtractMin	DecreaseKey	Dijkstra
Simple Array	$O( V )$	$O(1)$	$O( V ^2)$
Binary Heap	$O(\log  V )$	$O(\log  V )$	$O( E  \cdot \log  V )$
Fibonacci Heap	$O(\log  V )$	$O(1)$ (amorized)	$O( E  +  V  \log  V )$

## 2.4 Minimum Spanning Tree

**Example 2.8** (Minimum Spanning Tree (MST)). Input: Connected, undirected graph  $G = (V, E, \omega)$ .

**Definition 2.9** (Spanning Tree).  $T \subset E$  is a **spanning tree** if  $|T| = |V| - 1$ ,  $G' = (V, T)$  is connected.

**Goal of MST** Find spanning tree  $T$  so that  $\omega(T) = \sum_{e \in T} \omega(e)$  minimized.

**Theorem 2.10** (Cayley Theorem). The number of spanning trees of  $n$ -vertex complete graph is  $n^{n-2}$

A **cut**  $(S, V - S)$  has a **cutset** of  $S = \{e = (u, v) : u \in S, v \notin S\}$ .

**Claim 2.4.1.** Any cycle  $C$  and cutset  $D$  has intersection  $|C \cap D|$  even.

**Fundamental Cycle:** Given  $G$  and spanning tree  $T \subset E$ , for each  $e \in E \setminus T$ , the unique cycle in  $T \cup \{e\}$  is called **Fundamental cycle**.

**Claim 2.4.2.** For a fundamental cycle  $C$  related with  $e$ ,  $\forall f \in C \cap T$ ,  $(T \cup \{e\}) \setminus \{f\}$  is also a spanning tree.

If  $T$  is MST, then  $\omega(e) \geq \omega(f)$ .

**Fundamental Cut:** Spanning tree  $T \subset E$ . For each  $f \in T$ ,  $T \setminus \{f\}$  has two connected components, whose cutset is called **fundamental cut**.

**Claim 2.4.3.**  $\forall e \in D \setminus T$ ,  $(T \cup \{e\}) \setminus \{f\}$  is a spanning tree.

If  $T$  is MST, then  $\omega(e) \geq \omega(f)$ .

**MST Algorithm** There are some rules. **Red rule:** Let  $C$  a cycle without red edges. Select an uncolored edge in  $C$  with max weight and color it red.

**Blue rule:** Let  $D$  be a cutset without blue edges. Select an uncolored edge in  $D$  with min weight and color it blue.

**Greedy Algorithm:** Apply red or blue rules in any order iteratively until all edges colored.

**Theorem 2.11.** *Greedy algorithm terminates and blue edges form MST.*

*Proof.* Observed that during the algorithm, blue edges always form a forest.  $\square$

**Invariant**  $\exists$  MST  $T^*$  s.t.  $T^*$  contains all blue edges and no red edges.

*Proof.* Proof by induction. If there is a MST  $T^*$  contains all blue edges no red edges now. If we apply blue rule, with cutset  $D$  and  $f \in D$  but  $f \notin T^*$ , then for fundamental cycle  $C$  of  $f$ ,  $\forall e \in C \cap T, \omega(e) \geq \omega(f)$ . Since  $C$  has even edges in the cutset by the claim,  $\exists e \in C \cap T$  s.t.  $e \in D$ , which contradicts the fact that  $f$  is the edge in cutset  $D$  with min weight.

The case that we apply red rule is similar.  $\square$

---

### Algorithm 2 Prim's Algorithm

---

- 1: Initialize  $S \leftarrow \{s\}$ .
  - 2: **while**  $n - 1$  times **do**
  - 3:   Choose  $e$  be the min weight edge in the cutset  $(S, V \setminus S)$
  - 4:   add  $e$  to  $T$ , another endpoint of  $e$  to  $S$ .
  - 5: **end while**
- 

**Remark 2.12.** It is compatible with the simple idea: Each time chooses the min weight edge. However, it is more powerful since we only need to do this process in the cutset.

It is similar to Dijkstra's Algorithm. So its time complexity is  $O(|E| + |V| \log |V|)$

**Remark 2.13.** The first step need time complexity  $O(|E| \log |E|)$ .

The second step need time complexity  $O(|E| \cdot \alpha(|V|))$  using **Union-Find** data structure.

---

**Algorithm 3** Kruskal's Algorithm

---

- 1: Consider edges in weight increasing order.
  - 2: Add each edge to  $T$  if not introducing a cycle.
- 

WLOG we can assume edge weights are distinct.

---

**Algorithm 4** Boruvka's Algorithm

---

- 1: **while**  $< (n - 1)$  blue edges **do**
  - 2:   Simultaneously apply blue rule to each blue component.
  - 3: **end while**
- 

**Claim 2.4.4.** WHILE loop iterates  $\leq O(\log |V|)$ .

So time complexity is  $O(|E| \log |V|)$ .

**Remark 2.14.** There is a "contraction View". For each step, we can view each component as a single point with edges to other components.

If the graph is **Planar Graph**, then  $|E| \leq O(V)$ . At the  $i$ -th WHILE iteration,  
 $|V_i| \leq \frac{|V|}{2^{i-1}}, |E_i| \leq O(|V_i|)$ .

So the time complexity is  $\sum_i O(|E_i|) \leq \sum_i O\left(\frac{|V|}{2^{i-1}}\right) \leq O(|V|)$  which is linear!

Using the contraction view, we can get another algorithm:

**Prim+Boruvka**

- Run Boruvka for  $k$  iterates.
- Run Prim on the contracted graph.

**Remark 2.15.**

For step 1, time complexity is  $k \cdot |E|$ .

For step 2, time complexity is  $|E| + \frac{|V|}{2^k} \cdot \log \frac{|V|}{2^k}$ .

So the total time complexity is  $k|E| + \frac{|V|}{2^k} \cdot \log \frac{|V|}{2^k}$ .

Choose  $k = \log_2 \log_2 |V|$ , it comes to  $(\log \log |V|) \cdot |E| + \frac{|V|}{\log_2 |V|} \cdot \log_2 |V| \leq O(|E| \log \log |V| + |V|)$ .

## 2.5 Minimum Arborescence

**Example 2.16** (Minimum Arborescence). Input: Directed  $G = (V, E)$ , source  $s \in V$  and weight  $\omega : E \rightarrow \mathbb{R}$ .

We want to find an **arborescence**  $T = (V, E)$  with root  $r$  of minimum total weight.

**Definition 2.17.** Given directed  $G = (V, E)$  and  $r \in V$ ,  $n = |V|$ ,  $m = |E|$ ,  $F \subset E$  is an **arborescence** if

- $F$  is a spanning tree if ignoring directions
- $\forall v \in V, \exists$  unique path  $r \rightarrow v$  in  $F$ .

Or equivalently,  $F$  has no directed cycles and every node  $v \neq r$  has a unique incoming edge.

For this problem, WLOG we can assume that the root  $r$  has no in-degree and assume  $\omega \geq 0$ .

For each  $v \neq r$ , let

$$\text{cheap}(v) = \operatorname{argmin}_{e=(u,v) \in E} \{\omega(e)\}$$

**Claim 2.5.1.** Let  $F = \{\text{cheap}(v) | v \neq r\}$ .  $F$  is arborescence  $\Rightarrow F$  is min-cost.

Define  $\omega_r(u, v) = \omega(u, v) - \omega(\text{cheap}(v))$ . Suffices to find the min-cost arborescence under  $\omega_r$ .

If  $F$  is not an arborescence, then  $\exists$  a directed cycle  $C$  with all edges of weight 0.

Using the contraction view, if we contract "0-cycle" and keep this process recursively. By taking degrees carefully we can easily confirm the legality of the contraction view. Then suffices to prove it is indeed the min-cost arborescence when we expand after.

**Theorem 2.18.** *The min-cost arborescence  $\tilde{F}$  when we apply contraction to 0-cycle is exactly the min-cost arborescence in the original graph after expanding.*

**Lemma 2.19.**  $\exists$  min-cost  $F^*$  s.t. only 1 edge in  $F^*$  entering  $C$ .

*Proof.* Our goal is to prove  $\omega_r(F) \leq \omega_r(F^*)$ .

Let  $F_C^* = F^* \cap (C \times C)$ . Then  $|F_C^*| = |C| - 1$ .

Apply  $C$ -contraction to  $F^* \setminus F_C^*$  we obtain an arborescence of  $\tilde{G}$ . (Easy to check)

So

$$\sum_{e \in F^* \setminus F_C^*} \omega_r(e) \geq \sum_{e \in \tilde{F}} \omega_r(e)$$

So  $\omega_r(F^*) \geq \omega_r(F)$  □

*Proof of Lemma.* Choose any  $v \in C$ .

Let  $(x, y) \in r \rightarrow v$  be the first edge entering  $C$ .

Delete the edge entering  $C \setminus \{y\}$  and add the edge of circle except the edge entering  $y$ .

Then it is an arborescence of less cost. □

## 3 Dynamic Programming

### 3.1 Weighted Interval Scheduling

**Example 3.1** (Weighted Interval Scheduling). Input:  $n$  jobs,  $\{(s_i, f_i), \omega_i\}_{i=1}^n$ . Want to find  $\sum \omega_{i_k}$  maximum.

To make the structure simpler, we WLOG assume  $s_1 \leq s_2 \leq \dots \leq s_n$ . We may

---

**Algorithm 5** Search( $i$ )

---

- 1:  $j \leftarrow \min \text{id} > i, s_j \geq f_i$ .
  - 2: Return  $\max\{\text{Search}(j) + \omega_i, \text{Search}(i + 1)\}$ .
- 

find that there is a lot of repetitive computation. We can record each Search( $i$ )

---

**Algorithm 6** Search – Memorization( $i$ )

---

- 1: If  $i > n$ , RETURN 0
  - 2: If  $i \neq \text{bottom}$ , RETURN  $F[i]$ .
  - 3:  $j(i) \leftarrow \min\{j | s_j \geq f_i\}$ .
  - 4:  $F[i] \leftarrow \max\{\text{Search} - M(j(i)) + \omega_i, \text{Search} - M(i + 1)\}$
  - 5: RETURN  $F[i]$
- 

It can be written as

$$\begin{cases} F[i] = \max\{F[j(i)] + \omega_i, F[i + 1]\} \\ F[n + 1] = 0 \end{cases}$$

Such an equation is called **Bellman Equation**. So Dynamic Programming is a method to solve the problem by finding the optimal solution of each subproblem. We sometimes need to record the optimal solution of each subproblem to avoid repetition.

## 3.2 Segmented Least Square

**Example 3.2** (Least Square). We have  $n$  points  $\{(x_i, y_i)\}_{i=1}^n$ . We want to find a line  $y = ax + b$  to minimize

$$\text{SSE} = \sum_{i=1}^n [y_i - (ax_i + b)]^2 \quad (3.1)$$

Actually,

$$\begin{cases} a = \frac{n \sum_i x_i y_i - (\sum_i x_i) (\sum_i y_i)}{n \sum_i x_i^2 - (\sum_i x_i)^2} \\ b = \frac{\sum_i y_i - a \sum_i x_i}{n} \end{cases}$$

**Example 3.3** (Segmented Least Square). Input:  $\{(x_i, y_i)\}_{i=1}^n, c > 0$ .

Goal: Minimize  $l = E + cL$  for piecewise line, where  $c$  is the **hyperparameter**,  $L$  is the number of the segments.

WLOG, assume  $x_1 < x_2 < \dots < x_n$ .

We can define its subproblem as

$$\text{OPT}[i] : \min \text{loss}$$

when in put is  $(x_1, y_1), \dots, (x_i, y_i)$ .

Find solution  $\text{OPT}[n]$ . The boundary condition is  $\text{OPT}[1] = \text{OPT}[2] = c$  and the **Bellman Equation** is

$$\text{OPT}[i] = \min_{1 \leq j \leq i} \{ \text{OPT}[j-1] + l_{ji} + c \}$$

## 3.3 Knapsack Problem

**Example 3.4** (Knapsack Problem). Input:  $n$  items,  $w_i, v_i$  for its weight and value. The capacity of knapsack is  $w$ .



If assume integral weight, then denote  $\text{OPT}[i, w]$  as the optimal total value when in put is first knapsack capacity is  $w$ .

The **Bellman Equation** is

$$\text{OPT}[i, w] = \begin{cases} \text{OPT}[i - 1, w] & w < w_i \\ \max\{\text{OPT}[i - 1, w], v_i + \text{OPT}[i - 1, w - w_i]\}, w \geq w_i \end{cases}$$

It has time complexity  $O(nw)$ , which is not a polynomial algorithm.

We can find another Value-Based DP: (Also assume integral values)

$\text{OPT}[i, v]$ : choose min weight items.

from item  $1, 2, \dots, i$  so that total value  $\geq v$ .

The final solution for maxmial  $v$  s.t.  $\text{OPT}[n, v] \leq w$ .

$$\text{OPT}[i, v] = \min \begin{cases} \text{OPT}[i - 1, v] \\ w_i + \text{OPT}[i - 1, (v - v_i)^+] \end{cases}$$

$$\text{OPT}[0, v] = \begin{cases} 0 & v = 0 \\ +\infty & v > 0 \end{cases} \quad \text{The time complexity is } O(n^2v).$$

Now we consider a  **$\alpha$ -approximation algorithm** that  $\text{ALG} \geq \alpha \cdot \text{OPT}$  for  $\alpha \in (0, 1]$ .

Let  $\varepsilon = 1 - \alpha$ .

---

#### Algorithm 7 Knapsack Problem

---

- 1: Assume WLOG  $w_i \leq W$  so that  $V \geq \text{OPT}$ .
  - 2: Set  $K = \frac{\varepsilon V}{n}$ . Let  $v'_i = \left\lceil \frac{v_i}{K} \right\rceil$
  - 3: Run value-based DP to find optimal solution  $T$  for  $I'$
  - 4: Return  $T$  as a solution to  $I$ .
-

It is a feasible solution and

$$\begin{aligned}
\sum_{i \in T} v'_i &= \text{OPT}(I') \\
&\geq v(S; I'), \forall \text{feasible } S \\
&\geq v(T^*, I') \\
&= \sum_{i \in T^*} v'_i \\
&= \sum_{i \in T^*} \left\lfloor \frac{v_i}{K} \right\rfloor \\
&\geq \sum_{i \in T^*} \left( \frac{v_i}{K} - 1 \right) \\
&\geq \frac{1}{K} \sum_{i \in T^*} v_i - n \\
&= \frac{1}{K} \text{OPT}(I) - n
\end{aligned}$$

So  $\text{ALG} \geq \sum_{i \in T} K \cdot v'_i \geq \text{OPT}(I) - nK \geq (1 - \varepsilon) \text{OPT}(I)$ .

The time complexity is  $O(n^2 V') = O(n^2 \frac{V}{K}) = O(n^3 \varepsilon^{-1})$ .

**Remark 3.5.** The time complexity depends on the accuracy  $\varepsilon$  instead of the maximum value  $V$  since the accuracy is based on scale.

In other words,  $\varepsilon^{-1}$  in time complexity represents not only accuracy but also the "size" of scale.

**Fully Polynomial-Time Approximation Scheme (FPTAS)**  $\forall \varepsilon, \exists (1 - \varepsilon)$ -approximation algorithm with time complexity  $f(n, \varepsilon) = \text{poly}(n, \frac{1}{\varepsilon})$ .

**PTAS** :  $\forall \varepsilon, \exists (1 - \varepsilon)$ -approximation in time  $f_\varepsilon(n) = \text{poly}(n)$ . For this algorithm, it is  $(n \cdot 2^{\frac{1}{\varepsilon}}, n^{\frac{1}{\varepsilon}})$ .

### 3.4 RNA Secondary Structure

**Example 3.6** (RNA Secondary Structure). RNA is a string  $b_1b_2 \cdots b_n$  where  $b_i \in \{A, C, G, U\}$ .

The secondary structure is what fold to form "base pairs" including:

$$U \cdots A, A \cdots U, C \cdots G, G \cdots C$$

Mathematically, second structure represented by set of base pairs  $S = \{(i, j)\}$ ,

\*)  $\forall (i, x) \in S, (b_i, b_x) \in \{U \cdots A, A \cdots U, C \cdots G, G \cdots C\}$

\*) no sharp turns:  $\forall (i, j) \in S, i < j - 4$ ,

\*) non-crossing:  $\forall (i, j), (k, l) \in S$ , cannot have  $i < k < j < l$ .

Goal: Maximize  $|S|$ .

A direct idea is to construct those subproblems:

$$\text{OPT}[i, j] = \max_{i \leq k < j-4} \begin{cases} \text{OPT}[i, j-1] & b_j \text{ not matched} \\ 1 + \text{OPT}[i, k-1] + \text{OPT}[k+1, j-1] & b_j \text{ matched with } b_k \end{cases}$$

$$\text{OPT}[i, j] = 0 \text{ when } i \leq j < i + 4$$

### 3.5 Sequence Alignment(Edit Distance)

**Example 3.7.** For a wrong-spelled word, what cost do we need to make it right, using the gap and mismatch.

Or what is its edit distance to the correct word.

Mathematically, for string  $(a_1 \cdots a_n), (b_1 \cdots b_m)$ , a matching  $M = \{(i, j)\}$  such that there is no  $(i_1, j_1), (i_2, j_2) \in M$  s.t.  $i_1 < i_2$  but  $j_2 < j_1$ . Define its cost

$$\text{cost}(M) = \sum_{(i,j) \in M} \alpha_{a_i b_j} + \sum_{i \in [n], i \text{ not in } M} 1 + \sum_{\substack{j \in [m] \\ j \text{ not in } M}} \delta$$

$\sum_{(i,j) \in M} \alpha_{a_i b_j}$  is the mismatch cost and  $\sum_{i \in [n], i \text{ not in } M} 1 + \sum_{j \in [m], j \text{ not in } M} \delta$  is the gap cost

Define  $\text{OPT}[i, j]$  is the edit distance between  $a_1 a_2 \cdots a_i$  and  $b_1 b_2 \cdots b_j$ .

$$\text{OPT}[i, j] = \min_{1 \leq k \leq j} = \begin{cases} \delta + \text{OPT}[i-1, j] & a_i \text{ not matched} \\ \alpha_{a_i b_k} + \delta \cdot (j - k) + \text{OPT}[i-1, k-1] & a_i \text{ matched with } b_k \end{cases}$$

However, for each case it can be divided into three cases:

$$\text{OPT}[i, j] = \min \begin{cases} \text{OPT}[i-1, j-1] + \alpha_{a_i b_j} \\ \text{OPT}[i-1, j] + \delta \\ \text{OPT}[i, j-1] + \delta \end{cases}$$

The question is, if we need to trace the matching process, the space complexity is  $O(nm)$ , too large.

Here we use binary search.

---

#### Algorithm 8 Binary Search

---

- 1: Compute  $A[j] = d[(0, 0) \rightarrow (\frac{n}{2}, j)]$  and  $B[j] = d[(\frac{n}{2}, j) \rightarrow (n, m)]$ ,
  - 2: find  $j^* = \arg\min_j A[j] + B[j]$ .
  - 3: Run the sub-process  $(0, 0) \rightarrow (\frac{n}{2}, j^*)$  and  $(\frac{n}{2}, j^*) \rightarrow (n, m)$
- 

The complexity is still  $O(nm) + \frac{1}{2}O(nm) + \cdots + \frac{1}{2^k}O(nm) = O(nm)$ .

## 3.6 Matrix Multiplication

**Example 3.8** (Matrix Multiplication). Consider  $M_1 \cdot M_2 \cdots M_k$  where  $M_i$  is a  $n_{i-1} \times n_i$  matrix.

We want to find the optimal multiplicative order such that the time cost is minimal.

Denote  $\text{OPT}[i, j]$  is the min from  $M_i$  to  $M_j$ .

Using the binary tree, consider the last multiplication

$$\text{OPT}[i, j] = \min_{i \leq l < j} \{ \text{OPT}[i, l] + \text{OPT}[l + 1, j] + n_{i-1} n_l n_j \}$$

## 4 Flow Network

### 4.1 Definition

**Example 4.1.** For directed graph  $G = (V, E, s, t, c)$  where  $s$  is the source and  $t$  is the sink.  $c : E \rightarrow \mathbb{R}_{\geq 0}$  is the capacity function.

The **st-flow** is  $f : E \rightarrow \mathbb{R}_{\geq 0}$  s.t.

1)  $\forall e \in E, f(e) \leq c(e).$

2)  $\forall v \in V \setminus \{s, t\}, \sum_{(u,v) \in E} f(u, v) = \sum_{(v,u) \in E} f(v, u),$  i.e. flow conservation.

$$\text{val}(f) = \sum_{(s,u) \in E} f(s, u) - \sum_{(u,s) \in E} f(u, s)$$

Our goal is to maximize  $\text{val}(f)$

An **st-cut** is a partition  $(A, B)$  of  $V$  such that  $s \in A, t \in B$ , the capacity

$$c(A, B) = \sum_{\substack{(u,v) \in E \\ u \in A, v \in B}} c(u, v)$$

**Claim 4.1.1.**  $\forall$  feasible flow  $f$  and st-cut  $(A, B)$ ,

$$\text{val}(f) \leq c(A, B)$$

**Residual Network** Given flow network  $G$ , feasible flow  $f$ , the residual network  $G_f(v, E_f, s, t, c_f)$  is for each  $e \in E$

$$c_f(e) = c(e) - f(e) + f(e^{\text{reverse}})$$

where  $u \rightarrow v$  is on the flow.

**Claim 4.1.2 (Weak Duality).**  $f'$  is a feasible flow in  $G_f$  if and only if  $f \oplus f'$  is feasible in  $G$ , where

$$(f \oplus f')(e) = f(e) + f'(e) - f'(e^{\text{reverse}})$$

An **augmenting path**  $P$  is an unsaturated  $s \rightarrow t$  path in  $G_f$ .

---

**Algorithm 9** Augment  $(f, P)$

---

```

1: Let  $\delta = \min_{e \in P} c_f(e)$ .
2: for  $e = (u, v) \in P$  do
3:   if  $e \in E$  then
4:      $f(e) \leftarrow f(e) + \delta$ 
5:   else
6:      $f(v, u) \leftarrow f(v, u) - \delta$ 
7:   end if
8: end for

```

---

Now we give the Ford-Fulkerson Algorithm.

**Theorem 4.2.** *If F-F algorithm terminates, it finds a max flow.*

---

**Algorithm 10** Ford-Fulkerson Algorithm

---

```
1:  $f \leftarrow 0$ 
2: while  $\exists$  augmenting path  $P$  in  $G_f$  do
3:   Augment( $f, P$ )
4: end while
5: return  $f$ 
```

---

**Claim 4.1.3.**  $\forall$  st-cut  $(A, B)$ , st-flow  $f$ , we have

$$\text{val}(f) = \sum_{\substack{u \in A, v \in B \\ (u,v) \in E}} f(u, v) - \sum_{\substack{u \in E, v \in A \\ (u,v) \in E}} f(u, v)$$

It proves the previous claim weak duality.

*Proof.*

$$\begin{aligned} \text{val}(f) &= \sum_{(s,v) \in E} f(s, v) - \sum_{(u,s) \in E} f(u, s) \\ &\quad + \sum_{\omega \in A - \{s\}} \left( \sum_{(u,w) \in W} f(u, w) - \sum_{(w,v) \in E} f(w, v) \right) \end{aligned}$$

□

*Proof of the Theorem 4.2.* Consider the residue graph  $G$ .

Denote  $A$  to be the set of nodes reachable from  $s$ .  $B = V \setminus A$ .  $t \in B$  since there is no path from  $s$  to  $t$ .

Then st-cut  $(A, B)$  has capacity  $c_f(A, B) = 0$ . So for  $u \in B, v \in A$ , since  $f(u, v) \neq 0 \Rightarrow c_f(v, u) > 0$ , we have  $c_f(v, u) = 0 \Rightarrow f(u, v) = 0$ .

$$\text{val}(f) = \sum_{\substack{u \in A, v \in B \\ (u,v) \in E}} f(u, v) - \sum_{\substack{u \in B, v \in A \\ (u,v) \in E}} f(u, v)$$

$$\begin{aligned}
&= \sum_{\substack{u \in A, v \in B \\ (u,v) \in E}} c(u,v) - 0 \\
&= c(A, B)
\end{aligned}$$

□

Now suffices to proof that the algorithm terminates.

**Lemma 4.3.** *If capacities are integral and less than  $c$ , then F-F terminates in  $O(nmC)$  time and returns an integral max flow.*

The lemma implies we should choose some proper path so that it will terminate fast.

Assume the integral capacities  $\leq C$  and  $G_f(\Delta)$  denoted as  $G_f$  with edges of capacities  $\geq \Delta$ .

---

**Algorithm 11** Capacity-Scaling Algorithm

---

```

1: Initiate  $f \equiv 0, \Delta \leftarrow \text{largest } 2^k \leq c$ .
2: while  $\Delta \geq 1$  do
3:   while  $\exists$  augmenting path  $P$  in  $G_f(\Delta)$  do
4:     Augment( $f, P$ )
5:   end while
6:    $\Delta \leftarrow \Delta/2$ 
7: end while

```

---

**Theorem 4.4.** *The C-S runs in time  $O(m^2 \log c)$  since the step 2 runs for  $O(m)$  iterations.*

**Lemma 4.5.** *Every time inner WHILE terminates, max-flow value is less than  $\text{val}(f) + m\Delta$ .*

**Corollary 4.6.** *Each inner WHILE iterates  $\leq 2m$ . The times complexity is  $O(m^2 \log C)$*



*Proof of Lemma.* We let  $A$  be all nodes reachable from  $s$  and  $B = S \setminus A$ .

$$\begin{aligned}
\text{val}(f) &= \sum_{e \in E \text{ from } A \text{ to } B} f(e) - \sum_{e \in E \text{ from } B \text{ to } A} f(e) \\
&> \sum_{e \in E \text{ from } A \text{ to } B} (c(e) - \Delta) - \sum_{e \in E \text{ from } B \text{ to } A} (\Delta) \\
&= c(A, B) - \sum_{e \in E \text{ between } A, B} \Delta \\
&\geq c(A, B) - m\Delta \\
&\geq \text{MaxFlow} - m\Delta
\end{aligned}$$

□

---

### Algorithm 12 Shortest Augment Path

---

```

Initiate  $f \leftarrow 0$ .
while  $\exists s \rightarrow t$  path in  $G_f$  do
    Find  $P : s \rightarrow t$  in  $G_f$  using least number of edges.
    Augment  $(f, P)$ .
end while

```

---

**Lemma 4.7.** *Length of the shortest augmenting path never decreases.*

**Lemma 4.8.** *After  $\leq m$  iterations, length of the shortest augmenting path strictly increases. Time complexity is  $O(nm^2)$*

*Proof.* Assume  $f \xrightarrow{\text{augment}(f,P)} f'$  Denote  $l(u), l'(u)$  as the length of the shortest  $s \rightarrow u$  path in  $G_f, G_{f'}$  respectively.

Our goal is to prove  $l(u) \leq l'(u)$ .

$l(u)$  determines "distance" to  $s$ .

Define the level graph as the set of all  $(u, v) \in E(G_f)$  such that  $l(u) + 1 = l(v)$ .

Call edges not belong to level graph as *back edge*.

**Observation** Consider any  $e \in E(G_{f'}) \setminus E(G_f)$ ,  $e$  must be a back edge in  $G_f$ .

Choose  $u$  such that  $l'(u) < l(u)$  and  $l'(u)$  minimized.

If  $(v, u)$  is the edge in the shortest path of  $G_{f'}$

$$l(v) \leq l'(v) = l'(u) - 1 < l(u) - 1 \leq l(u) - 2$$

so  $(u, v)$  is not a back edge in  $G_f$ , hence  $(v, u) \notin E(G_{f'})$ , which causes contradiction.  $\square$

**Lemma 4.9.** After  $\leq m$  augmentation,  $\exists u$ ,  $l(u)$  strictly increases. It goes on no more  $n^2$  times, so the time complexity is  $O(n^2 m^2)$ .

*Proof.* This lemma is much easier than the previous lemma.

Noticed that each augmentation adds back edges and removes at least one edges in level graph.  $\square$

*Proof of Lemma 4.8.*

**Claim 4.1.4.** During the period when  $l(t)$  doesn't increase, the added edges in residual graph does not appear in shortest augmenting path.

Suppose for contradiction:  $\exists j < i$ ,  $l_j(t) = l_i(t)$ ,  $\exists (v, u)$  appears in the shortest augmenting path  $P$  in  $G_{f_i}$  and  $l_j(v) = l_j(u) + 1$ .

Choose the edge  $(v, u)$  with smallest  $i$  and then with largest  $l_i(u)$ .

Then  $l_i(u) \geq l_j(u) + 2$ . So

$$\begin{aligned} l_j(t) &\leq l_j(u) + |P[u \rightarrow t]| \\ &\leq l_i(u) + |P[u \rightarrow t]| - 2 \\ &\leq l_i(t) - 2 \end{aligned}$$

**Recent work** [Chen et al. '2022] we can do in  $O(m^{1+o(1)})$ .

## 4.2 Applllication

### 4.2.1 Bipartite Matching

**Example 4.10** (Bipartite Matching). For Bipartite graph  $G = (U, V, E)$ , a matching  $M \subset E$ , we want to find  $M$  to maximize  $|M|$ .

We can construct two virtual nodes  $s, t$  such that  $s \rightarrow$  all nodes in  $U$  and  $t \rightarrow$  all nodes in  $V$ , with capacity 1.

Then the maximum capacity of flow in the augmented graph is what we need.

**So the meaning of capacity can be generalized as the number of one node who can accommended**

Now consider so-called "perfect matching", *i.e.*  $|M| = |U| = |V|$ .

Note that  $\exists$  perfect matching *s.t.*  $\forall S \subset U, |\Gamma(S)| \geq |S|$ .

**Theorem 4.11** (Hall's Theorem). *The inverse still holds. i.e. If  $\forall S \subset U, |T(S)| \geq |S|$ , then  $\exists M$  is perfect matching if  $|M| = n$ .*

*Proof.* It suffices to prove  $\text{max-flow} = n$ . Or equivalently, to prove  $\text{min-cut} \geq n$ .

*i.e.*  $\forall s\text{-}t \text{ cut } (A, B), c(A, B) \geq n$ .

$c(A, B) < +\infty \xRightarrow{1}$  if  $u \in A$  then  $\Gamma(u) \in A \Rightarrow \Gamma(A \cap U) \subset A \cap V$ .

$$\begin{aligned} c(A, B) &\geq |B \cap U| + |A \cap V| \\ &\geq n - |A \cap U| + |\Gamma(A \cap U)| \\ &\geq n \end{aligned}$$

**Remark 4.12.** Here we mark the weight between  $U$  and  $V$  to be  $\infty$  such that the fact 1 holds.

The duality of max-flow and min-cut is very useful in this problem.

### 4.2.2 Network Connectivity

**Example 4.13** (Network Connectivity). Directed  $G = (V, E)$ , source  $s$ , sink  $t$ . Then Max-flow = the maximum number of edge-disjoint  $s \rightarrow t$  path. Two paths are called *edge-disjoint* if they have no edge in common.

Connectivity of the graph is defined as the  $\min_{E' \subset E} |E'|$  such that  $s \rightarrow t$  disconnected in  $(V, E \setminus E')$

**Theorem 4.14** (Menger's Theorem). *Connectivity = min-cut = max-flow = maximum number of edge-disjoint  $s \rightarrow t$  path.*

### 4.2.3 Circulation

**Example 4.15.** Directed graph  $G = (V, E)$  with capacity  $c : E \rightarrow \mathbb{R}_{\geq 0}$  and node demand  $d : V \rightarrow \mathbb{R}$ . ( $d(u) < 0$  means the supply node)

We have the flow conservation

$$\sum_{e \text{ into } u} f(e) - \sum_{e \text{ out of } u} f(e) = d(u), \forall u$$

Our task is to decide whether there exists a feasible flow  $f$  satisfies the flow conservation.

Indeed, we can construct two virtual nodes  $s, t$  such that  $s$  to all nodes with demand  $d < 0$ , equipped with capacity  $d$  and  $t$  has edges from all nodes with demand  $d > 0$ , equipped with capacity  $d$ .

Then the task is equivalent to check whether the max-flow saturates all edges out of  $s$  and in of  $t$ .

Moreover, we can use the cut to discuss.

We have:

$$\nexists \text{ feasible circulation} \Leftrightarrow \exists \text{ cut } (A, B) \text{ s.t. } c(A, B) < \sum_{v \in B} d(v).$$

**Remark 4.16.** This criterion, similar as Hall's Theorem 4.11, is so-called "*polynomial proof*", under the meaning that for a specific case, we can give a proof in polynomial time to check.

**Flow Lower Bounds** If we have a capacity constraint such that

$$l(e) \leq f(e) \leq c(e), \forall e \in E$$

Then it suffices to add the lower flow at first. For example, for two nodes with demand 0 and edge with amount in  $[4, 6]$ , we can replace it with

$$4 \xrightarrow{[0,6]} -4$$

#### 4.2.4 Survey Design

**Example 4.17** (Survey Design). We ask  $n_1$  customers about  $n_2$  products. Ask customer  $i$  the number between  $[c_i, c'_i]$  products and ask the number between  $[p_j, p'_j]$  customers questions about product  $j$ .

We want to find if there is a feasible survey design.

It is equivalent to give each edge a weight interval, where  $s \rightarrow i$  with  $[c_i, c'_i]$ ,  $i \rightarrow j$  with  $[0, 1]$  and  $j \rightarrow t$  with  $[p_j, p'_j]$ .

### 4.2.5 Airline Scheduling

**Example 4.18** (Airline Scheduling). Flight  $i$  from the origin  $o_i$  at time  $s_i$  to the destination  $d_i$  at time  $f_i$ .

We want to know what is the minimum number of crews in flights that can be scheduled. A feasible schedule for one crew is a set of flights  $\{i_1, i_2, \dots\}$  such that  $f_{i_k} \leq s_{i_{k+1}}, d_k = o_{k+1}$ .

We can construct a graph with nodes  $o_i \rightarrow d_i$ . The edges  $o_i \rightarrow d_i$  with weight  $[1, 1]$ . If the schedule from flight  $i$  to flight  $j$  is feasible *i.e.*  $f_i \leq s_j$ , we let edge  $i \rightarrow j$  with weight interval  $[0, 1]$ .

Then a feasible flow gives a feasible schedule. To limit the total amount, we can determine the minimum number.

**Remark 4.19.** The weight interval have a broader meaning in this problem. With different view of nodes and edges, we can transform it into different limitations.

### 4.2.6 Image Segmentation

**Example 4.20** (Image Segmentation). For an image,  $p_{ij} \geq 0$  is the separation penalty if neighbors  $i, j$  belongs to different partitions.

$a_i \geq 0$  is the likelihood that  $i \in A$  (foreground)  $b_i \geq 0$  is the likelihood that  $i \in B$  (background)

Our goal is to partition pixels into  $A, B$ , to maximize

$$\sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{\substack{i, j \text{ neighbors} \\ |\{i, j\} \cap A| = 1}} p_{ij}$$

It is equivalent to

$$\begin{aligned} & \text{minimize} \quad - \sum_{i \in A} a_i - \sum_{j \in B} b_j + \sum_{\substack{i,j \text{ neighbors} \\ |\{i,j\} \cap A|=1}} p_{ij} \\ \Leftrightarrow & \text{minimize} \quad \sum_{i \in B} a_i \sum_{j \in A} b_j + \sum_{\substack{i,j \text{ neighbors} \\ |\{i,j\} \cap A|=1}} p_{ij} \end{aligned}$$

Then we can construct a visual source with edges to all pixels with weight  $a_i$  and a visual sink with edges from all pixels with weight  $b_i$ . All neighbors of pixel have edges of weight  $p_{ij}$  from each other

**Remark 4.21.** This example focuses on the optimal sum of net flow. To construct visual source, sink and proper edges, we can optimize some sum of structure with related constraints.

#### 4.2.7 Project Selection

**Example 4.22** (Project Selection).  $v \rightarrow w$ ,  $v$  depends on  $w$ . Our goal is to find a feasible set  $S$  of projects (if  $v \in S$ , then all prerequisites of  $v \in S$ ) to maximize

$$\sum_{v \in S} p(v)$$

- Introduce the virtual source node  $s$  and the virtual sink node  $t$ .
- Assign a capacity of  $\infty$  to each prerequisite edge.
- Add edge  $(s, v)$  with capacity  $p(v)$  if  $p(v) > 0$ .
- Add edge  $(v, t)$  with capacity  $-p(v)$  if  $p(v) < 0$

Then the min-cut  $(A, B)$  satisfies:

1)  $\forall (u, w) \in E, u \in S \Rightarrow w \in A$

2)

$$\begin{aligned}
 c(A, B) &= \sum_{\substack{v \in B \\ p(v) > 0}} p(v) + \sum_{\substack{v \in A \\ p(v) < 0}} (-p(v)) \\
 &= \sum_{v: p(v) > 0} p(v) - \sum_{\substack{v \in A \\ p(v) > 0}} p(v) - \sum_{\substack{v \in A \\ p(v) < 0}} p(v) \\
 &= \sum_{v: p(v) > 0} p(v) - \sum_{v \in A} p(v)
 \end{aligned}$$

So it suffices to compute  $c(A, B)$ !

**Remark 4.23.** Use edges of capacity  $\infty$ , we can reduce some situation we do not want.

#### 4.2.8 Baseball Elimination

**Example 4.24.** Given set of team  $S$ , distinguished team  $z \in S$ . Team  $x$  has won  $w_x$  games already. Team  $x$  and  $y$  play each other  $r_{xy}$  additional games.

Given the current standings, is there any outcome of the remaining games in which team  $z$  finishes with the most (or tied for the most) wins?

Assume team  $z$  wins all remaining games.  $M = w_z + \sum_x r_{zx}$ .

We want to arrange the remaining games that do not involve team  $z$  so that all other teams have  $\leq M$  wins.

- Construct two virtual nodes  $s$  and  $t$ .
- Assign a capacity of  $\infty$  to the edge from the match  $i - j$  to team  $i$  and  $j$ .
- Assign a capacity of  $M - w_i$  to the edge from team  $i$  to  $t$ .



- Assign a capacity of  $r_{ij}$  to the edge from team  $s$  to the match  $i - j$ .

Then  $z$  can possibly win the most games iff the max-flow saturates for all edges from  $s$ .

**Certificate of Elimination**  $T \subset \{2, 3, \dots, n\}$ ,  $\omega(T) = \sum_{i \in T} \omega_i$ ,  $r(T) = \sum_{i < j \in T} r_{ij}$ ,  
 $\frac{\omega(T) + r(T)}{|T|} > M$ .

Noticed that if the following equation holds, then team  $z$  is theoretically eliminated.

$$\frac{\omega(T) + r(T)}{|T|} > M \quad (4.1)$$

**Theorem 4.25.** Team  $q$  is theoretically eliminated iff  $\exists T \subset \{2, 3, \dots, n\}$  s.t. (4.1) holds

*Proof.* If  $\forall T \subset \{2, 3, \dots, n\}$ ,  $\frac{\omega(T) + r(T)}{|T|} < M$ , then

$$\min - \text{cut} \geq \sum_{1 < i < j \leq n} r_{ij} = r(\{2, 3, \dots, n\})$$

That's because, if we consider any  $s - t$  cut  $(A, B)$  such that  $c(A, B) < +\infty$ .

1)  $i \in B \Rightarrow \text{match } i - j \in B, \forall j$ .

$$\begin{aligned} c(A, B) &\geq \sum_{i \in A} (M - w_i) + \sum_{\substack{i \in B \text{ or } j \in B \\ 1 < i < j \leq n}} r_{ij} \\ &= r(\{2, 3, \dots, n\}) + |A \setminus \{s\}| \cdot M - \omega(A \setminus \{s\}) - r(A \setminus \{s\}) \\ &\geq r(\{2, 3, \dots, n\}) \end{aligned}$$

The inverse is trivial. □

**Remark 4.26.** Note that we use the duality of max-flow and min-cut in this problem.

## 5 Introduction to Approximation Algorithm

There are plenty of NP-hard optimization problems.

**Example 5.1 (3-CNF).** For  $n$  Boolean variables  $\{x_1, x_2, \dots, x_n\}$ .

A *literal* is either a value  $x_i$  or negative value  $\bar{x}_i$ .

A *clause* is a disjunction of literals. *e.g.*  $(x_1 \vee x_2 \vee \bar{x}_3)$ .

A **CNF** is the conjunction of clauses. *e.g.*  $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$ .

A **3-CNF** is a CNF with at most 3 literals in each clause.

**Decision problems** Check whether we can choose  $x_1, \dots, x_n$

For maximization problems,  $A$  is an  $\alpha$ -**approximation algorithm** if

$$\text{Val}(A(I), I) \geq \alpha \cdot \text{OPT}(I), \alpha \in (0, 1]$$

For minimization problem, it will be

$$\text{Val}(A(I), I) \leq \alpha \cdot \text{OPT}(I), \alpha \in [1, \infty)$$

Max-cut problem: Find the maximal number of bichromatic edges using 2 colors. We have

$$|\text{edge}(A, B)| = \sum_{i=1}^n \max\{a_i, b_i\} \geq \sum_{i=1}^n \frac{a_i + b_i}{2} = \frac{1}{2}|E| \geq \frac{1}{2}\text{OPT}$$

So this is a  $\frac{1}{2}$ -approximation algorithm for max-cut problem.

However, if we consider it as a minimization problem, the value can be 0, so the scale can be  $+\infty$ . Hence, it is different if we consider the approximation algorithm of min-uncut or max-cut problems.

Decision Problems	Natural Optimization Problems
Is there a truth assignment that satisfies the 3-CNF formula?	Maximize the number of clauses satisfied by a truth assignment.
3-coloring (NP-complete)	<ul style="list-style-type: none"> <li>• Min-coloring</li> <li>• Max-3-cut: Max number of bichromatic edges using 3 colors.</li> <li>• Min-3-Uncut: Min number of monochromatic edges using 3 colors.</li> </ul>
2-coloring (P complete)	<ul style="list-style-type: none"> <li>• Max-cut</li> <li>• Min-Uncut</li> </ul>
Vertex Cover: Given $G = (V, E), k$ . Decide whether $\exists$ Vertex-Cover using $\leq k$ vertices.	Min-vertex-Cover: Given $G = (V, E)$ . Find $S \subset V$ s.t. $\forall e = (u, v) \in E,  \{u, v\} \cap S  \geq 1,  S $ is minimized.

Table 1: Comparison of Decision Problems and Natural Optimization Problems

**c vs. s Decision Problem** Decide whether  $\text{OPT}(I) \geq c$  or  $\text{OPT}(I) < s$ . If  $\text{OPT}(I) \geq c$ , return YES, else if  $\text{OPT}(I) < s$ , return NO.

**c vs. s Decision Problem** Given  $I$  s.t.  $\text{OPT}(I) \geq c$ , find a solution  $x$  s.t.  $\text{Val}(x; I) \geq s$ .

**Theorem 5.2.** Suppose  $A$  solves  $c$  vs.  $s$  Search problem in poly-time, then  $\exists$  poly-time  $A'$  that solves  $c$  vs.  $s$  decision problem.

The algorithm is as follows:

---

**Algorithm 13 Greedy**

---

```
initiate  $A, B \leftarrow \emptyset$ 
for  $i$  from 1 to  $n$  do
  Let  $a_i \leftarrow$  number of edges between  $A$  and  $i$ .
  Let  $b_i \leftarrow$  number of edges between  $B$  and  $i$ .
  if  $a_i < b_i$  then
     $A \leftarrow A \cup \{i\}$ .
  else
     $B \leftarrow B \cup \{i\}$ 
  end if
end for
```

---

---

**Algorithm 14  $c$  vs.  $s$  Search**

---

```
1:  $x \leftarrow A(I)$ 
2: if  $\text{Val}(x; I) \geq s$  then
3:   return YES
4: else
5:   return NO
6: end if
```

---

**Fact 5.3.**

- (1)  $A$  is  $\alpha$ -approximation algorithm  $\Rightarrow A$  is  $c$  vs.  $\alpha c$  search algorithm  $\forall c$ .
- (2)  $\exists c$  vs.  $s(c)$  search algorithm  $\Rightarrow \exists \alpha$ -approximation algorithm where  $\alpha = \inf_c \left\{ \frac{s(c)}{c} \right\}$
- (3) (contrapositive of A)  $c$  vs.  $s$  decision problem "hard"  $\Rightarrow \frac{s}{c}$ -approximation algorithm "hard".

**Remark 5.4.** The same  $c$  vs.  $s$  algorithm in max-cut and min-uncut problems might correspond to quite different approximation ratios.

## 5.1 Set-Cover

**Example 5.5 (Set-Cover).** Universe  $U = \{1, 2, \dots, n\}$ .  $S_1, S_2, \dots, S_M \subset U$ .

Our goal is to find  $T \subset \{S_1, \dots, S_M\}$  such that  $\cup_{S \in T} S = U$  and  $|T|$  minimized.

Of course, it can be represented as **Max-Coverage** problem: Given additional input  $k$ . Find  $T \subset \{S_1, \dots, S_M\}$  s.t.  $|T| = k$  and  $|\cup_{S \in T} S|$  maximized.

It has a greedy algorithm:

---

**Algorithm 15 Greedy**

---

```

1:  $T \leftarrow \emptyset$ 
2: repeat
3:   Let  $S_i$  be the set that covers the most uncovered elements.
4:    $T \leftarrow T \cup \{S_i\}$ .
5:    $U \leftarrow U \setminus S_i$ .
6: until  $\begin{cases} \text{All elements covered} & \text{set cover} \\ |T| = k & \text{max-coverage} \end{cases}$ 

```

---

**Fact 5.6.** Suppose  $\exists m$  sets covering  $U$ . After  $t$  choices,  $T$  covers  $1 - \left(1 - \frac{1}{m}\right)^t$  fraction of elements.

**Corollary 5.7.** The greedy algorithm is  $\lceil \ln n \rceil$  approximation for set-cover

*Proof.* Let  $m = \text{OPT}$ . After  $t = \lceil \ln n \rceil \cdot m$  choices, number of uncovered elements

$$\left(1 - \frac{1}{m}\right)^{\lceil \ln n \rceil \cdot m} \cdot n \geq \frac{1}{n} = 1$$

□

**Corollary 5.8.** Greedy is  $1$  vs.  $1 - \frac{1}{e}$  approximation for Max-coverage.

*Proof.* Set  $m = k$ . After  $t = k$  choices, coverage of  $T$ :

$$1 - \left(1 - \frac{1}{k}\right)^k \geq 1 - \frac{1}{e}$$

□

**Fact 5.9.**  $1$  vs.  $1 - \gamma$  approximation for  $ma$ -coverage  $\Rightarrow \lceil \log_{1-\gamma} \frac{1}{n} \rceil$ -approximation for set-cover.

*Proof.* "Guess"  $k = \text{OPT}^{\text{set-cover}}$ .

Repeatedly invoke  $A(k) \cdot \lceil \log_{1-\gamma} \frac{1}{n} \rceil$  times. Then number of uncovered elements

$$n \cdot (1 - \gamma)^{\lceil \log_{1-\gamma} \frac{1}{n} \rceil} \geq n \cdot \frac{1}{n} = 1$$

□

In fact, we can construct an extreme case for greedy algorithm:

## Algorithmic Gap of Greedy for Min-Set-Cover (cont'd)

• **More refined hard instances.** For any integer  $c \geq 2$  and  $n \gg c$ , we construct the sets as below:

$S_1, S_2, \dots, S_c$  partition  $U$  into equal parts.

$S_{c+1}$  includes the first  $1/c$  fraction elements of  $S_1, S_2, \dots, S_c$ ,

$S_{c+2}$  includes the first  $1/c$  fraction elements of  $S_1 - S_{c+1}, S_2 - S_{c+1}, \dots, S_c - S_{c+1}$ ,

$S_{c+3}$  includes the first  $1/c$  fraction elements of  $S_1 - S_{c+1} - S_{c+2}, \dots, S_c - S_{c+1} - S_{c+2}$ ,

... till  $S_{c+\alpha}$  where  $\alpha = \log_{1-1/c} \frac{c}{n}$

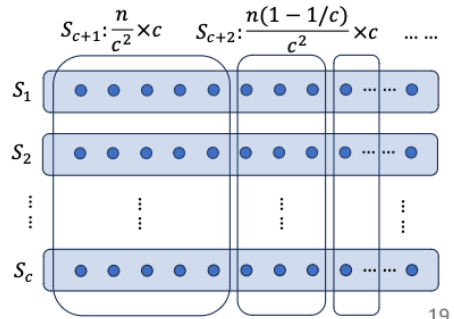
• **Analysis.** The optimal solution only chooses  $S_1, S_2, \dots, S_c$ .

Assuming Greedy breaks ties in the worst way,

Greedy may sequentially choose  $S_{c+1}, \dots, S_{c+\alpha}, S_1, \dots, S_c$ .

• Greedy gap ratio for this instance is:

$$\frac{c+\alpha}{c} = 1 + \frac{\ln n/c}{c \ln c/(c-1)} \sim \ln n \text{ (when } c \rightarrow \infty \text{)}.$$



19

Figure 5.1: Extreme case for greedy algorithm

## 5.2 Weighted Min Set-cover and Randomized Rounding

**Example 5.10** (Weighted Min Set-cover). Given  $n$  elements,  $M$  sets,  $S_1, \dots, S_M \subset U$ . Each set  $S_i$  has a weight  $w(S_i) > 0$ .

Now select  $T \subset \{S_1, \dots, S_M\}$  such that  $\sum_{S \in T} w(S)$  minimized.

Integer Program: Minimize  $\sum_{i=1}^M w(S_i)x_i$ . Subject to  $\sum_{i:j \in S_i} x_i \geq 1, \forall j \in U$ , where  $x_i \in \{0, 1\}, \forall i \in [M]$ .

If we relax the integer constraint, we have an LP relaxation:  $x_i \in [0, 1]$ , which can be solved in poly-time since it is a linear program.

We need "rounding" to transform fractional solution to the integer solution.

### 5.2.1 Randomized Rounding

If  $\{x_i^*\}$  is the optimal LP solution. For each  $s_i$ , select  $s_i \in T$  independently with probability  $\min\{\alpha x_i^*, 1\}$ . Then

$$\mathbb{E}[w(T)] \leq \alpha \sum_{i=1}^M w(s_i), \quad x_i^* = \alpha \cdot \text{LP} \leq \alpha \cdot \text{OPT}$$

Now we want to estimate  $\Pr[T \text{ covers } U]$ .

If there is some  $\alpha x_i^* \geq 1$ , then  $\Pr(T \text{ covers } U) = 1$ . If  $\forall i, x_i^* < 1$ , then

$$\begin{aligned} \Pr[T \text{ covers } U] &= 1 - \Pr[\exists j \in U, j \notin T] \\ &\geq 1 - \sum_{j \in U} \Pr[j \notin T] \\ &= 1 - \sum_{j \in U} \prod_{i:j \in S_i} (1 - \min\{\alpha x_i^*, 1\}) \\ &\geq 1 - \sum_{j \in U} \prod_{i:j \in S_i} \exp(-\alpha x_i^*) \quad (\text{If some } x_i^* < 1, \text{ the probability will be } 0) \\ &= 1 - \sum_{j \in U} \exp(-\sum_{i:j \in S_i} \alpha x_i^*) \\ &= 1 - \sum_{j \in U} \exp(-\alpha) \end{aligned}$$

Therefore, we obtain

**Claim 5.2.1.**

$$\Pr(\text{every element covered}) \geq 1 - n \cdot e^{-\alpha}$$

If we set  $\alpha = \ln n + \ln \ln n$ , then

$$\begin{aligned} \Pr[\varepsilon_1] \Pr[\text{every element covered}] &\geq 1 - n \cdot e^{-\ln n - \ln \ln n} \\ &= 1 - \frac{1}{n \ln n} \\ &\geq 1 - \frac{1}{\ln n} \end{aligned}$$

We should also focus on  $\mathbb{E}(w(T)) \leq \alpha \cdot \text{OPT}$ . Here we use the **Markov Inequality**:

**Theorem 5.11** (Markov Inequality). *For Random Variable  $X \geq 0$ ,  $\Pr(X \geq t\mathbb{E}X) \leq \frac{1}{t}$*

*Proof.*

$$\begin{aligned} \mathbb{E} X &= \mathbb{E}[X|X \geq \alpha] \cdot \Pr(X \geq \alpha) + \mathbb{E}[X|X < \alpha] \cdot \Pr(X < \alpha) \\ &\geq \alpha \cdot \Pr[X \geq \alpha] \end{aligned}$$

Therefore,  $\Pr[X \geq \alpha] \leq \frac{\mathbb{E} X}{\alpha}$

□

So

$$\begin{aligned} \Pr[\varepsilon_2] \Pr \left[ \sum_i w(S_i) y_i \geq (\ln n + 2 \ln \ln n) \text{OPT} \right] &\leq \frac{\ln n + \ln \ln n}{\ln n + 2 \ln \ln n} \\ &= 1 - \frac{\ln \ln n}{\ln n + 2 \ln \ln n} \end{aligned}$$



So the union probability

$$\begin{aligned}
\Pr[\varepsilon \wedge \bar{\varepsilon}_2] &\geq \Pr[\varepsilon_1] - \Pr[\varepsilon_2] \\
&\geq \frac{\ln \ln n}{\ln n + 2 \ln \ln n} - \frac{1}{\ln n} \\
&\geq \Omega\left(\frac{\ln \ln n}{\ln n}\right) \xrightarrow{\text{boost}} 1 - \frac{1}{n^{100}}
\end{aligned}$$

**Theorem 5.12.** *With probability  $\Omega(\frac{\ln \ln n}{\ln n})$ , LP+randomized rounding returns a  $(\ln n + 2 \ln \ln n)$ -approximation solution.*

The question is how to boost the probability. Indeed, if we independently round  $N$  times,

$$\begin{aligned}
\Pr[\exists 1 \text{ trial succeeds}] &\geq 1 - [1 - \Omega(\frac{\ln \ln n}{\ln n})]^N \\
&\geq 1 - \exp(-\Omega(\frac{\ln \ln n}{\ln n} \cdot N)) \\
&\geq 1 - \exp(-\Omega(n \cdot \ln n)) \geq 1 - e^{-n}
\end{aligned}$$

if we set  $N \leftarrow (\ln n) \cdot n$  in the last step.

Apply the method to Max-coverage problem, Integer Problem is to  $\max \sum_{j=1}^n y_j$  such that

$$\begin{aligned}
\sum_{i=1}^M x_i &\leq k \\
y_j &\leq \sum_{i \in S_j} x_i, \forall j \in [n] \\
y_j &\leq 1, \forall j \in [n] \\
x_i &\in \{0, 1\}, \forall i \in [M]
\end{aligned}$$

So the LP relaxation is to relax  $x_i \in [0, 1]$ .

Repeat  $k$  times and select set  $i$  with probability  $\frac{x_i^*}{k}$ .

$$\begin{aligned}
\mathbb{E}[\text{coverges}] &= \sum_{j=1}^n \Pr[j \text{ covered}] \\
&= \sum_{j=1}^n \left( 1 - \left( 1 - \sum_{i \in S_j} \frac{x_i^*}{k} \right)^k \right) \\
&\geq \sum_{j=1}^n \left( 1 - \exp\left(- \sum_{i \in S_j} x_i^*\right) \right) \\
&\geq \sum_{j=1}^n (1 - \exp(-y_j^*)) \\
&\geq \alpha \sum_{j=1}^n y_j^* = \alpha \cdot \text{LP} \geq \alpha \cdot \text{OPT}
\end{aligned}$$

where  $\alpha = 1 - \frac{1}{e}$

The estimation is tight for this rounding if we consider the set  $\mathcal{U} = \{0, 1\}^k$  and  $S_{i,0} = \{a \in \mathcal{U} | a_i = 0\}$ ,  $S_{i,1} = \{a \in \mathcal{U} | a_i = 1\}$ .

### 5.2.2 Integrality Gap

Instance  $I$  is a  $c$  vs.  $s$ -**Integrality Gap (IG) instance** if  $\text{LP}(I) \geq c$  and  $\text{OPT}(I) \leq s$ . The **gap ratio** is  $\frac{c}{s}$ .

1. Large IG  $\Rightarrow$  Inaccurate estimation of LP.
2. The estimation of rounding algorithm is usually

$$\text{rounding} \geq \dots \geq \alpha \cdot \text{LP} \geq \alpha \cdot \text{OPT}$$

Since  $\text{rounding} \leq \text{OPT}$  in maximization problem, if IG is large,  $\alpha$  can be very large and hence the approximation ratio is very bad.

For instance, consider the set-cover problem that is to minimize  $\sum_{i=1}^M x_i$  s.t.

$$\sum_{j \in S_i} x_j \geq 1, \forall j \in U, \quad x_i \in [0, 1], \forall i \in [M]$$

relax the condition  $x_i \in [0, 1]$  and consider the set  $\mathcal{U} = \{0, 1\}^q \setminus \{0\}$  and  $S_{\vec{\alpha}} = \{l \in U : l^T \alpha = 1 \pmod{2}\}$  for  $\alpha \in \{0, 1\}^q$  with the size  $M = 2^q, n = 2^q - 1$ .

$$|S_{\vec{\alpha}}| = \begin{cases} 2^{q-1} & \alpha \neq 0 \\ 0 & \alpha = 0 \end{cases}$$

**Claim 5.2.2.** LP=2

*Proof.* Take  $x_{\vec{\alpha}} = \frac{2}{2^q}$ . Then  $\sum_{\vec{\alpha}} S_{\vec{\alpha}} = 2$ . And the LP constraint met where

$$\forall \vec{e} \in U, \sum_{\vec{\alpha}} \frac{2}{2^q} = 2 \Pr_{\vec{\alpha} \in \{0,1\}^q} [\vec{e} \in S_{\vec{\alpha}}] = 2 \times \frac{1}{2} = 1$$

Certainly LP  $\geq 2$ , so LP = 2. □

But we also have a claim about OPT:

**Claim 5.2.3.** OPT  $\geq q$ . So the instance is a 2 vs.  $q$  IG with ratio  $\frac{q}{2} = \frac{1}{2} \log_2(n+1) = \frac{\ln(n+1)}{2 \ln 2}$ .

*Proof.* For any  $S_{\vec{\alpha}_1}, \dots, S_{\vec{\alpha}_{q-1}}$ , suppose  $S_{\vec{\alpha}_1} \cup \dots \cup S_{\vec{\alpha}_{q-1}}$  is a cover of  $U$ . Then

$$\begin{aligned} &\Leftrightarrow \bar{S}_{\vec{\alpha}_1} \cap \dots \cap \bar{S}_{\vec{\alpha}_{q-1}} = \{\vec{0}\} \\ &\Leftrightarrow \{\vec{e} \in \{0, 1\}^q : e^T \vec{\alpha}_i = 0 \forall i \in [q-1]\} = \{\vec{0}\} \end{aligned}$$

which is impossible! □

$I$  is an  $\alpha$ -Integrality Gap instance if

$$\text{LP}(I) \leq \frac{1}{\alpha} \cdot \text{OPT}(I)$$

Then we indeed prove that  $\alpha$  can be  $\frac{1}{2} \log_2(n+1) < \ln(n+1)$  in Min-Set-Cover problem.

Take  $U = \{1, 2, \dots, n\}$ .  $M$  are  $C \cdot k \ln n$  sets, each of which  $s_i$  includes each  $j \in U$  independently with probability  $\frac{1}{k}$

**Claim 5.2.4.** When  $C \geq \frac{4}{\varepsilon^2}$ , the probability

$$\Pr[\text{LP} \leq \frac{k}{1-\varepsilon}] \geq 1 - \frac{1}{n}$$

*Proof.* Consider  $x_i = \frac{k}{1-\varepsilon}$ .

Fix  $j \in [n]$ .

$$\begin{aligned} \Pr\left[\sum_{j \in S_i} x_i \geq 1\right] &= \Pr\left[\sum_{i=1}^M \mathbf{1}[j \in S_i] \geq \frac{(1-\varepsilon)M}{k}\right] \\ &\geq 1 - \left[\frac{e^{-\varepsilon}}{(1-\varepsilon)^{1-\varepsilon}}\right]^{\frac{M}{k}} \\ &= 1 - \exp\left(\left(-\varepsilon - (1-\varepsilon) \ln(1-\varepsilon)\right) \cdot \frac{M}{k}\right) \\ &\geq 1 - \exp\left(-\frac{\varepsilon^2}{2} \cdot \frac{M}{k}\right) \\ &\geq 1 - \exp(-2 \ln n) \end{aligned}$$

Here we use the Chernoff bound with a high relation of central limit theorem.

**Theorem 5.13** (Chernoff Bound).  $X_1, X_2, \dots, X_n \in [0, 1]$  a.s. and  $\mathbb{E} X_i = p_i$ . Let

$X = X_1 + \cdots + X_n, \mathbb{E} X = \mu$ . For any  $\delta > 0$

$$\begin{cases} \Pr[X \geq (1 + \delta)\mu] \leq \left[ \frac{e^\delta}{(1+\delta)^{1+\delta}} \right]^\mu \\ \Pr[X \leq (1 - \delta)\mu] \leq \left[ \frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \right]^\mu \end{cases}$$

□

**Claim 5.2.5.** For  $k \geq \frac{2}{\varepsilon}$  and  $n = n(k, \varepsilon, C)$  large enough, we have

$$\Pr[\text{OPT} \geq (1 - \varepsilon)k \ln n] \geq 0.99$$

*Proof.* Let  $z = (1 - \varepsilon)k \ln n$ .

$\text{OPT} > z \Leftrightarrow \forall \mathcal{S} \in \binom{[M]}{z}, \mathcal{S} \text{ doesn't cover } U$ . We consider probability of the latter case.

Now fix  $\mathcal{S} \in \binom{[M]}{z}$ ,  $\Pr[\mathcal{S} \text{ cover } U]$  is actually

$$\begin{aligned} \Pr[\mathcal{S} \text{ cover } U] &= \Pr[\forall j \in U, \exists S_i \in \mathcal{S}, j \in S_i] \\ &= \Pr[\exists S_i \in \mathcal{S}, 1 \in S_i]^n \\ &= \left( 1 - \left( 1 - \frac{1}{k} \right)^z \right)^n \\ &\leq \exp\left(-n \left( 1 - \frac{1}{k} \right)^z\right) \\ &\stackrel{k \geq 2}{\leq} \exp\left(-n \exp\left(-\left(1 - \varepsilon\right)\left(1 + \frac{1}{k}\right) \ln n\right)\right) \\ &\stackrel{k \geq \frac{2}{\varepsilon}}{\leq} \exp\left(-n \exp\left(-\left(1 - \frac{\varepsilon}{2}\right) \ln n\right)\right) \\ &= \exp\left(-n \cdot n^{-(1-\frac{\varepsilon}{2})}\right) \\ &= \exp\left(-n^{\frac{\varepsilon}{2}}\right) \end{aligned}$$

So

$$\begin{aligned}
\Pr[\exists \mathcal{S} \in \binom{[M]}{z}, \mathcal{S} \text{ cover } U] &\leq \binom{M}{z} \cdot \exp(-n^{\frac{\varepsilon}{2}}) \\
&\leq \left( \frac{C \cdot e}{1 - \varepsilon} \right)^{(1-\varepsilon)k \ln n} \cdot \exp(-n^{\frac{\varepsilon}{2}}) \\
&< \exp(-n^{\frac{\varepsilon}{4}}) \\
&< 0.01
\end{aligned}$$

as  $n$  large enough. Here we end the proof □

So using Randomized construction,  $\alpha$  can approach  $(1 - \varepsilon) \ln n$  for any  $\varepsilon$ .

## 5.3 Hardness of Approximation

### 5.3.1 P, NP classes

For  $\mathcal{L} \in \{0, 1\}^*$  is the 0–1 encoding, a problem is the set of some 0–1 encoding and a decision problem is to decide whether  $\mathcal{L}$  belongs to it.

For instance,  $\mathcal{L}_k$  is the set of all (0–1 encoding) of set cover instances where  $U$  can be covered by  $k$  sets. We define

$$P = \{\mathcal{L} : \mathcal{L} \text{ can be poly-time decided by a (deterministic) Turing machine}\}$$

$$NP = \{\mathcal{L} : \mathcal{L} \text{ can be poly-time decided by a non-deterministic Turing machine}\}$$

NP problems are all problems that can be "verified" in poly-time. Explicitly,

for input instance  $x \in \{0, 1\}^*$ , the prover is based on  $x$ , providing a "proof"  $y \in \{0, 1\}^*$  that  $|y| \leq \text{poly}(|x|)$ , however, the verifier is a poly-time algorithm that accepts  $x, y$  and outputs YES/NO.

In other words,  $\mathcal{L} \in NP \Leftrightarrow \exists$  a prover-verifier system such that

- Completeness:  $\forall x \in \mathcal{L}, \exists \text{ proof } y \text{ s.t. verifier returns YES in poly-time.}$
- Soundness:  $\forall x \notin \mathcal{L}, \forall \text{ proof } y, \text{ verifier returns NO in poly-time.}$

The equivalence is because we actually can "guess" the proof  $y$  in a non-deterministic TM.

If  $P=NP$ , then if we can verify proof in poly-time, we can also construct it in poly-time. There isn't innovation anymore!

**NP-complete:**  $\mathcal{L}$  is NPC if

- 1)  $\mathcal{L} \in NP$
- 2)  $\forall \mathcal{L}' \in NP, \mathcal{L}' \leq_p \mathcal{L} \text{ i.e. } \mathcal{L}' \text{ can be reduced to } \mathcal{L} \text{ in poly-time.}$

Equivalently, if some NPC problems can be solved in poly-time, then  $P=NP$ .

If only 2) in the definition of NPC holds, then it is a **NP-hard** problem.

We define the **polynomial reduction**  $M \leq_p L$  if

$\exists$  poly-time algorithm  $A$  such that  $\forall x \in \{0, 1\}^*$ ,

- (completeness)  $x \in M \text{ returns YES} \Rightarrow A(x) \in L \text{ returns YES.}$
- (soundness)  $x \in M \text{ returns NO} \Rightarrow A(x) \in L \text{ returns NO.}$

Observed that if  $M$  is NP-Complete and  $M \leq_p \mathcal{L}$ , then  $L$  is NP-Hard.

**Theorem 5.14** (Cook-Levin). *3-SAT is NP-Complete.*

*Proof.*  $\forall L \in NP$ , need to show  $L \leq_p 3\text{-SAT}$ .

Let  $A$  be the poly-time verifier (DTM) for  $L$ .

Now we consider the original DFA, which needs start, process and after, denoted as  $(s, p, \alpha)$

For time  $t$ , the tape can be

$$t_{-M}^{(t)}, \dots, t_M^{(t)}, \alpha^{(t)}, S^{(t)}, p^{(t)}$$

where the transition function is

$$t_i^{(\tau)} = g_i(t_i^{(\tau-1)}, \alpha^{(\tau-1)}, s^{(\tau-1)}, p^{(\tau-1)})$$

$$s^{(\tau)} = h(\alpha^{(\tau-1)}, s^{(\tau-1)}, p^{(\tau-1)})$$

$$p^{(\tau)} = \dots$$

$$\alpha^{(\tau)} = \dots$$

which is a compose of bool function. So any DFA process can be converted to a 3-SAT instance.

□

**Theorem 5.15** (Max-Coverage). *Deciding whether Max-Coverage=100% is NP-Complete.*

*Proof.* We divide it into two parts:

1. Max-coverage=1 is NP.
2. 3-SAT  $\leq_p$  Max-coverage=1.

Consider any 3-SAT instance  $I$ . We have variables  $x_1, \dots, x_n$  and clauses  $c_1, \dots, c_m$ .

Denote  $U = \{x_1, \dots, x_n, c_1, \dots, c_m\}$  and sets  $S_1, S_2, \dots, S_n, S_{n+1}, \dots, S_{2n}$ . For  $i = 1, 2, \dots, n$ ,

$$S_i = \{x_i\} \cup \{c_j : c_j \text{ contains } x_i\}$$

$$S_{n+i} = \{x_i\} \cup \{c_j : c_j \text{ contains } \bar{x}_i\}$$

Let  $k = n$ .

Completeness: If  $I$  satisfiable, then  $\exists \sigma : \{x_i\} \rightarrow \{0, 1\}$ , choose



$$\begin{cases} S_i & \text{if } \sigma(x_i) = 1 \\ S_{n+i} & \text{if } \sigma(x_i) = 0 \end{cases}$$

$$\text{Soundness: If } J \text{ is YES, for } I, \text{ let } \sigma(x_i) = \begin{cases} 1 & \text{if } S_i \text{ chosen} \\ 0 & \text{if } S_{i+n} \text{ chosen} \end{cases}.$$

□

Now we want to consider the approximation problem.

1 vs. 1 Max-Coverage is NP-H but what if decide the gap-version  $s$  vs.  $c$ .

**Observation** If we could prove  $c$  vs.  $s$  M-C is NP-H for  $c > s$ . Then  $\frac{s}{c}$  approximation M-C problem is NP-H.

**Theorem 5.16** (PCP theorem).  $\exists \varepsilon$  s.t.  $\text{Max} - 3 - \text{SAT}_{1,1-\varepsilon}$  is NP-Hard.

We give an introduction for PCPs.

**Definition 5.17** (Probability Checkable Proofs). Verifier: input instance  $x$  and proof  $y$ .

Reads  $x$ , compute a (joint) distribution  $D$  over the locations in  $y$ , and a Boolean function.

Sample  $i, j, k, f \sim D$ .

output YES iff  $f(y_i, y_j, y_k) = 1$ .

- (completeness) If  $x$  is YES, then  $\exists y$  s.t.  $\Pr[\text{Verifier accepts}] \geq c$ .
- (soundness) If  $x$  is NO, then  $\forall y, \Pr[\text{Verifier accepts}] \leq s$ .

Then PCP theorem is equivalent to

**Theorem 5.18** (PCP theorem).  $\exists \varepsilon > 0$  s.t. every NP problem has a PCP system with  $c = 1, s = 1 - \varepsilon$ .

**Definition 5.19.**  $\text{PCP}_{c,s}[r, q]$  denotes set of languages that admits a PCP system with  $c, s, r, q$  parameters. Explicitly,

- Prover reads input, outputs poly-length proof with unbounded computational prover.
- Verifier in poly-time reads input and  $r$  random bits, (deterministically by input and random bits) computes  $q$  locations in the proof, reads the  $q$  bits in the proof and decides YES/NO.
- The systems satisfies completeness and soundness:
  - (Completeness) Input is YES instance  $\Rightarrow \exists \text{ proverPr}[\text{Verifier accepts}] \geq c$ .
  - (Soundness) Input is NO instance  $\Rightarrow \forall \text{ proverPr}[\text{Verifier accepts}] \leq s$ .

**Observation 5.20.**

- $\text{PCP}_{1, \frac{1}{2}}[0, 0] = P$ .
- $\text{PCP}_{1, \frac{1}{2}}[0, \text{poly}(n)] = NP$ .
- $\text{PCP}_{1, \frac{1}{2}}[O(\log(n)), O(1)] \leq NP$

For the final observation, we actually can construct a Verifier to enumerate all possible random bits in  $\{0, 1\}^r$  to return YES if there is some possibility larger than  $c$ .

Indeed, PCP theorem is actually,

**Theorem 5.21** (PCP theorem).

$$\text{PCP}_{1, \epsilon}[O(\log n), O(1)] = \text{PCP}_{1, \frac{1}{2}}[O(\log n), O(1)] = NP$$

**Proposition 5.22.** *PCP theorem  $\Leftrightarrow \exists s < 1$ ,  $\text{Gap} - 3\text{MAXSAT}_{1,s}$  is NP-Hard.*

*Proof.* " $\Rightarrow$ ": Our goal is to prove  $3\text{SAT} \leq_p \text{Gap} - 3\text{MAXSAT}_{1,s}$ , i.e. given a 3-SAT instance  $\phi$ , we can construct an instance  $\Phi$  in poly-time such that  $\text{OPT}(\phi) = 1 \Rightarrow \text{OPT}(\Phi) = 1$ .

$3\text{SAT} = \text{GAP} - 3\text{MAXSAT}_{1, \frac{1}{m}}$  is NP-hard. By PCP theorem,  $\exists$  a prover-verifier system for 3-SAT that with  $c = 1$ ,  $s = \frac{1}{2}$ ,  $r = O(\log n)$ ,  $q = O(1)$

Prover provides proof  $\vec{x} \in \{0, 1\}^N$ .

Given  $\phi$ ,  $\forall \vec{\tau} \in \{0, 1\}^r$ , verifier computes

$$l_1, \dots, l_q \in \{1, 2, \dots, N\}$$

$$f : \{0, 1\}^q \rightarrow \{0, 1\}$$

find 3-CNF  $g$  over  $q + q \cdot 2^q$  variables  $\{z_{\vec{\tau}}\}$  and  $q \cdot 2^q$  clauses  $c_{\vec{\tau}}$  such that  $f(\vec{y}) = 1$  iff  $\exists \vec{z} \in \{0, 1\}^{q \cdot 2^q}$ ,  $g(\vec{y}, \vec{z}) = 1$ .

Construct  $\Phi$  with variables  $\{x_1, x_2, \dots, x_N\} \cup \bigcup_{\tau} z_{\tau}$  and clauses  $\bigwedge_{\tau} c_{\tau}$ .

Completeness: If  $\exists \vec{x}$  such that  $\Pr_{\vec{\tau}}[\text{Verifier accepts}] = 1$ . Then  $\forall \vec{\tau}$ ,  $\exists z_{\vec{\tau}}$  such that  $c_{\vec{\tau}} = 1$ .

Soundness:  $\forall \vec{x}$ ,  $\Pr_{\vec{\tau}}[\text{Verifier accepts}] \leq \frac{1}{2}$ . Consider a solution  $\sigma$  to  $\Phi$ . Let  $T = \{\vec{\tau} : \text{Verifier rejects } \sigma(X) \text{ under } \vec{\tau}\}$ . Then  $|T| \geq \frac{1}{2} \cdot 2^r$ .

$\forall \vec{\tau}$ ,  $\sigma$  doesn't satisfy all clauses in  $C_{\vec{\tau}}$  so the number of unsatisfied clauses  $\geq |T| = \frac{1}{2} \cdot 2^r$ .

$$\text{val}(\sigma : \Phi) \leq 1 - \frac{|T|}{2^r \cdot q \cdot 2^q} = 1 - \frac{\frac{1}{2} \cdot 2^r}{2^r \cdot q \cdot 2^q} = 1 - \frac{1}{2q \cdot 2^q}$$

" $\Leftarrow$ ":  $\forall$  NP language  $\mathcal{L} \leq_p \text{GAP} - 3\text{SAT}_{1,s}$ . Then

$$\mathcal{L} \in \text{PCP}_{1,s}[O(\log n), 3] \leq \text{PCP}_{1, \frac{1}{2}}[O(\log(n)), O(1)]$$

**Theorem 5.23.**  $\forall \varepsilon > 0$ ,  $\text{Gap} - 3\text{SAT}_{1, \frac{7}{8} + \varepsilon}$  is NP-Hard.

**Corollary 5.24.**  $\forall \varepsilon > 0$ ,  $(\frac{7}{8} + \varepsilon)$ -approximation Max3SAT is NP-Hard.

The corollary is equivalent to  $\forall \varepsilon > 0$ ,  $\text{Gap}3\text{SAT}_{1-\varepsilon, \frac{7}{8} + \varepsilon}$  is NP-Hard.

**Remark 5.25.** It implies that it is hard to find an algorithm better than random algorithm. It also shows that perfect completeness is sometimes very hard.

## 5.4 Label-Cover Games

To prove the theorem, we need to consider a constraint Graph  $G = (U, V, E)$ , which is a bipartite graph.

Prover is a function  $\sigma : U \rightarrow [K], V \rightarrow [L]$ .

Constraints: For each  $e = (u, v) \in E$ ,  $\pi_e : [L] \rightarrow [K]$ .

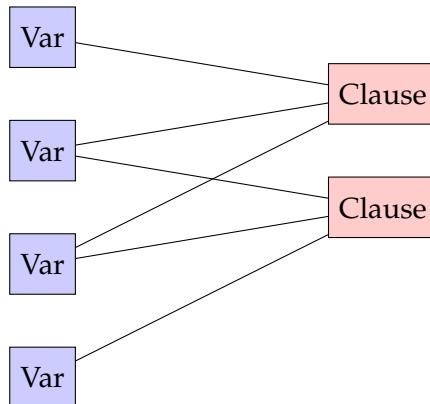
Verifier: Uniformly sample  $e = (u, v) \in E$ , accepts only if  $\pi_e(\sigma(v)) = \sigma(u)$ .

The system is also called "2-Prover-1Verifier Game" or "Projection Game".

More generally,  $\pi_e \subset [K] \times [L]$ .

**Claim 5.4.1.** PCP Theorem implies that  $\exists \delta > 0$ ,  $\text{Gap} - \text{LC}_{1, 1-\delta}^{(K, L)=(2, 7)}$  is NP-Hard.

*Proof.* Reduce from  $\text{Gap} - 3\text{SAT}_{1, 1-\varepsilon}$ .



Variables  $x_i$  have  $\sigma(x_i) \in \{0, 1\}$  and Clauses  $c_i = x_{j_i}^1 \wedge x_{j_i}^2 \wedge x_{j_i}^3$  have  $\sigma(c_i) \in [7]$  to represent the state of  $c_i$ . □

### 5.4.1 Paralled Repetition

Given  $H = (G(U, V, E), K, L, \{\pi_e\})$ .

$$H^{\otimes t} = (G^{\otimes t}, K^t, L^t, \{\pi_{e_1, e_2, \dots, e_t}\})$$

where

$$G^{\otimes t} = \{(u_1, u_2, \dots, u_t), (v_1, v_2, \dots, v_t)\}$$

$$\pi_{(u_1, v_1), \dots, (u_t, v_t)} = \{((\alpha_1, \dots, \alpha_t), (\beta_1, \dots, \beta_t)) : (\alpha_i, \beta_i) \in \pi_{(u_i, v_i)}\}$$

It is easy to check that if  $H$  is a projection game,  $H^{\otimes t}$  is also a projection game.

We wonder whether the following theorem holds

**Theorem 5.26** (not quite true).

$$\text{OPT}(H^{\otimes t}) \leq \text{OPT}(H)^t$$

There is a counter-example for  $U = \{u_1, u_2\}, V = \{v_1, v_2\}, K = L = 4$  and  $G$  is fully connected.  $[K], [L] \leftrightarrow \{u, v\} \times \{1, 2\}$

$$\pi_{(u_i, v_j)} = \{((u, i), (u, i)), ((v, j), (v, j))\}$$

Clearly,  $\text{OPT}(H) = \frac{1}{2}$ .

However,  $\text{OPT}(H^{\otimes 2}) = \frac{1}{2}$ .

Let  $\sigma((u_{i_1}, u_{i_2})) = (((u, i_1), (v, i_1))), \sigma((v_{j_2}, v_{j_2})) = ((u, j_1), (v, j_2))$ .

So verifier accepts if  $i_1 = j_2$ .

However, the following theorem holds:

**Theorem 5.27.** Suppose  $H$  has alphabet size less than  $k$  and  $\text{OPT}(H) \leq 1 - \delta$ . Then

$$\text{OPT}(H^{\otimes t}) \leq \exp(-\Omega(\frac{\delta^3 t}{\log K}))$$

**Corollary 5.28.**  $\forall \delta > 0, \exists K, L$  such that  $\text{GAP} - \text{LC}(K, L)_{1,\delta}$  is NP-Hard.

In fact, the above hardness still holds even when the constraints graph is regular and (and  $|U| = |V|$ )

Recall max-coverage problem is:  $U = \{1, 2, \dots, n\}, S_1, S_2, \dots, S_m \subset U, k \leq m$ . Our goal is to select  $k$  sets to maximize  $\frac{1}{|U|} |\text{union of selected sets}|$ .

We have proved that  $(1 - \frac{1}{e})$ -approximation is NP-hard.

**Theorem 5.29.**  $\forall \varepsilon > 0, \text{Gap} - \text{MC}_{1, \frac{3}{4} + \varepsilon}$  is NP-Hard.

*Proof.* Suffices to reduce LC case to MC case.

Let  $W = \bigcup_{(u,v) \in E} T_{uv} \times \{(u, v)\}, |W| = 2^k \cdot |E|$ , where  $T_{uv} = \{0, 1\}^k$ .

Denotes

$$T_{uv, \alpha, 0} = \{\vec{x} : x_\alpha = 0\}$$

$$T_{uv, \pi_{uv}(\beta), 1} = \{\vec{x} : x_{\pi_{uv}(\beta)} = 1\}$$

Let

$$S_{u, \alpha} = \bigcup_{(u,v) \in E} T_{uv, \alpha, 0} \times \{(u, v)\}, \forall u \in U, \alpha \in [K]$$

$$S_{v, \beta} = \bigcup_{(u,v) \in E} T_{uv, \pi_{uv}(\beta), 1} \times \{(u, v)\}, \forall v \in V, \beta \in [L]$$

Take  $k = |U| + |V|$ .

**Completeness:**  $\exists \sigma$  satisfies all constraints in LC. Then select  $\{S_{u, \sigma(u)}, S_{v, \sigma(v)}\}$  to achieve 100% coverage.

**Soundness:** If  $\text{OPT}(LC) < \delta, \Rightarrow \text{OPT}(MC) < \frac{3}{4} + \varepsilon$ .

Otherwise, if  $\exists$  set selection achieving  $\frac{3}{4} + \varepsilon$  coverage, then one can "decode" a  $\sigma$  such that  $\text{Val}(\sigma; \text{LC}) \geq \delta$ .

Let

$$\text{Sugg}(u) = \{\alpha, S_{u,\alpha} \text{ selected}\}, \text{Sugg}(v) = \{\beta : S_{v,\beta} \text{ selected}\}$$

**Claim 5.4.2.**

$$\mathbb{E}_{(u,v) \sim E} [|\text{Sugg}(u)| + |\text{Sugg}(v)|] = 2$$

*Proof of Claim.*

$$\begin{aligned} \mathbb{E}_{(u,v) \sim E} [|\text{Sugg}(u)| + |\text{Sugg}(v)|] &= \sum_{(u,v) \in E} \frac{1}{|E|} (|\text{Sugg}(u)| + |\text{Sugg}(v)|) \\ &= \mathbb{E}_{u \sim U} |\text{Sugg}(u)| + \mathbb{E}_{v \sim V} |\text{Sugg}(v)| \\ &= \frac{1}{|U|} \left( \sum_u |\text{Sugg}(u)| + \sum_v |\text{Sugg}(v)| \right) \\ &= 2 \end{aligned}$$

□

Here we use Corollary 5.28 with the stronger version that the graph is regular.

Decoding Scheme:  $\forall u \in U$ , choose  $\sigma(u)$  uniformly from  $\text{Sugg}(u)$ ,  $\forall v \in V$ , choose  $\sigma(v)$  uniformly from  $\text{Sugg}(v)$ .

**Definition 5.30.** Edge  $(u, v) \in E$  is *consistently suggested* if  $\exists \alpha \in \text{Sugg}(u), \beta \in \text{sugg}(v)$  such that  $\pi_{uv}(\beta) = \alpha$ .

**Fact 5.31.** If  $(u, v)$  is consistently suggested, then

$$\Pr_{\sigma}[(u, v) \text{ satisfied}] \geq \frac{1}{|\text{Sugg}(u)| \cdot |\text{Sugg}(v)|}$$

Now consider

$$E_1 = \{(u, v) \in E \mid (u, v) \text{ consistently suggested}\}$$

$$E_0 = E \setminus E_1, \quad \gamma = \frac{|E_1|}{|E|}$$

**Lemma 5.32.** *If  $(u, v) \in E_0$ , then*

$$\text{coverage of } T_{uv} \leq 1 - 2^{-(|\text{Sugg}(u)| + |\text{Sugg}(v)|)}$$

*Proof.* Note that if  $|\text{Sugg}(u)| = |\text{Sugg}(v)| = 1$ , then coverage of  $T_{uv} \leq \frac{3}{4}$ ,

$$\begin{aligned} & \text{non-coverage of } T_{u,v} \\ &= \Pr_{\vec{x} \sim \{0,1\}^k} [\forall \alpha \in \text{Sugg}(u) : x_\alpha = 1 \vee \forall \beta \in \text{Sugg}(v) : x_{\pi(\beta)} = 0] \\ &= \Pr_{\vec{x} \sim \{0,1\}^k} [\forall \alpha \in \text{Sugg}(u) : x_\alpha = 1] \cdot \Pr_{\vec{x} \sim \{0,1\}^k} [\forall \beta \in \text{Sugg}(v) : x_{\pi(\beta)} = 0] \\ &= 2^{-|\text{Sugg}(u)|} \cdot 2^{-|\pi(\text{Sugg}(v))|} \\ &\geq 2^{-(|\text{Sugg}(u)| + |\text{Sugg}(v)|)} \end{aligned}$$

□

**Definition 5.33.** Edge  $(u, v)$  is  $\tau$ -good if  $\min\{|\text{Sugg}(u)|, |\text{Sugg}(v)|\} \leq \tau$ . Then if  $(u, v) \in E_1$  and  $\tau$ -good, then  $\Pr_\sigma[(u, v) \text{ satisfied by } \sigma] \geq \frac{1}{\tau^2}$

Let

$$\mathbb{E}_{(u,v) \sim E_1} [|\text{Sugg}(u)| + |\text{Sugg}(v)|] = \tau$$

Then at least  $\frac{1}{2}$  edges in  $E_1$  are  $(2\tau)$ -good.



Then

$$\mathbb{E}_\sigma [\text{Val}(\sigma; LC)] \geq \gamma \cdot \frac{1}{2} \cdot \frac{1}{(2\tau)^2}$$

For the original Max-coverage problem, it subjects to

$$\begin{aligned} \frac{3}{4} + \varepsilon &\leq \gamma \cdot 1 + (1 - \gamma) \left[ 1 - \mathbb{E}_{(u,v) \sim E_0} 2^{-|\text{Sugg}(u,v)|} \right] \\ &\leq \gamma + (1 - \gamma) \left[ 1 - 2^{-\mathbb{E}_{(u,v) \sim E_0} |\text{Sugg}(u,v)|} \right] \\ &= \gamma + (1 - \gamma) \left[ 1 - 2^{-\frac{2-\gamma\tau}{1-\gamma}} \right] \end{aligned}$$

Suffices to prove a claim that

**Claim 5.4.3.** If  $\gamma + (1 - \gamma) \left[ 1 - 2^{-\frac{2-\gamma\tau}{1-\gamma}} \right] \geq \frac{3}{4} + \varepsilon$ , then

$$\gamma \geq \frac{4}{1 + \ln 4} \varepsilon > \varepsilon$$

and

$$\tau \leq \frac{2}{\varepsilon}$$

So

$$\gamma \cdot \frac{1}{8\tau^2} \geq \varepsilon \cdot \frac{\varepsilon^2}{32} = \frac{\varepsilon^3}{32} > \delta$$

Here we prove that

$$\text{Gap} - \text{MC}_{1, \frac{3}{4} + \varepsilon} \xleftarrow{\delta = \varepsilon^3/32} \text{Gap} - \text{LC}(K, L)_{1, \delta}$$

□

## 5.5 Multicut

**Example 5.34** (Multicut). Input: undirected graph  $G = (V, E)$ , weights  $\omega : E \rightarrow \mathbb{R}_{\geq 0}$  and terminal pairs  $\{(s_i, t_i)\}_{i=1}^k$ .

Our goal is to find  $E' \subset E$  to minimize  $\omega(E') = \sum_{e \in E'} \omega(e)$  such that  $s_i, t_i$  disconnected in  $(V, E - E')$ ,  $\forall i \in [k]$ .

Clearly,  $k = 1$  is the min-cut problem.

**Theorem 5.35.**  $k = 2$  can be solved in polynomial time.

**Fact 5.36.**  $k \geq 3$ , then  $k$ -approximation problem is easy. Actually, we can consider the union of all min  $(s_i, t_i)$ -cut.

**Example 5.37** (Vertex Cover). Input  $G = (V, E)$ .

Our goal is to select  $V' \subset V$  to minimize  $|V'|$  such that  $\forall e \in E$ ,  $e$  has  $\geq 1$  endpoints in  $V'$ .

Those two problems are equivalent. Acutally,  $\text{OPT}(I) = \text{OPT}(J)$  for two corresponding instances.

**Theorem 5.38.** 1.414-approximation for VC is NPH. Assuming UG-Conjecture,  $(2-\varepsilon)$ -approximation is hard.

As a result, 1.414-approximation for multicut is NPH. Assuming UGC, no poly-time constant approximation for multi-cut.

The goal in this lecture is to give a  $O(\log n)$ -approximation approximation.

### 5.5.1 Multicut on Trees

Consider LP relaxation:

$$\min \sum_{e \in E} \omega(e) \cdot \chi_e$$

$$\begin{aligned}
s.t. \quad & \sum_{e \in P(s_i, t_i)} \chi_e \geq 1 \\
& \chi_e \geq 0, \forall e \in E
\end{aligned}$$

where  $P(s_i, t_i)$  is the unique path on tree.

**Rounding** Let  $d(v) = \sum_{e \in P(r, v)} \chi_e, \forall v \in V$ .

Sample  $\theta \in [0, \frac{1}{2})$  uniformly.

Say  $\theta$  cuts  $e = (u, v)$  if  $[d(u), d(v)) \cap \{\theta, \theta + \frac{1}{2}, \theta + 1, \theta + \frac{3}{2}, \dots\} \neq \emptyset$ .

Let  $E' = \{e \in E \text{ cut by } \theta\}$ .

Feasibility: Consider any  $(s_i, t_i)$ . Let  $u_i$  be the least common ancestor of  $s_i, t_i$ .

Then

$$d(s_i) - d(u_i) + d(t_i) - d(u_i) \geq 1$$

Assume WLOG  $d(s_i) - d(u_i) \geq \frac{1}{2}$ . Then  $P(s_i, u_i) \cap E' \neq \emptyset$ .

Quality: Consider  $e = (u, v)$ ,

$$\Pr[e \text{ cut by } \theta] = \begin{cases} 1 & \chi_e \geq \frac{1}{2} \\ 2\chi_e & \chi_e < \frac{1}{2} \end{cases}$$

$$\mathbb{E}_\theta \omega(E') = \sum_e \Pr[e \text{ cut by } \theta] \cdot \omega(e) \leq \sum_e 2\chi_e \omega(e) = 2\text{LP}$$

## 5.5.2 Multicut on General Graphs

LP relaxation:

$$\begin{aligned}
& \min \sum_{e \in E} \omega(e) \cdot \chi_e \\
s.t. \quad & \sum_{e \in P(s_i, t_i)} \chi_e \geq 1, \forall P \text{ connect } s_i, t_i
\end{aligned}$$

$$\chi_e \geq 0, \forall e \in E$$

**Theorem 5.39.** *LP poly-time solvable if  $\exists$  poly-time "separation oracle"*

**Definition 5.40.** Given  $\{\chi_e\}$ , if  $\{\chi_e\}$  is feasible, then oracle returns "YES". Otherwise, oracle returns "NO" and any one of the violated constraint.

**Theorem 5.41** (Low-Diameter Decomposition, LDD). *Given  $G = (V, E, \omega)$ , metric space  $(V, d)$ ,  $D > 0$ .  $\exists$  partition of  $V = S_1 \cup S_2 \cup \dots \cup S_t$  such that*

- *Low Diameter:*  $\forall i \in [t]$ , *diameter of  $s_i \leq D$ .*
- *Low cutting cost:*

$$\sum_{e \in E \text{ cut by the partition}} \omega(e) \leq \frac{O(\log n)}{D} \sum_{e=(u,v) \in E} \omega(e) \cdot d(u, v)$$

If the theorem holds, we can let  $d(u, v)$  be the shortest path distance w.r.t.  $\{\chi_e\}$ ,  $D = 0.99$ ,  $E'$  be the set of edges cut by the partition. Here we construct an  $O(\log n)$ -approximation algorithm.

Then

$$\omega(E') \leq \frac{O(\log n)}{D} \sum_{e=(u,v) \in E} \omega(e) \chi_e = \frac{O(\log n)}{D} \text{LP}$$

*Proof of Theorem 5.41.* Construct partition  $\{S_v\}_{v \in V}$ . Let  $r = \frac{D}{2}$  be the radius.

## Algorithm

1. Sample  $X \sim \text{Unif}[\frac{r}{2}, r]$ .
2. Uniformly randomly order vertices in  $V$ .
3. For each vertex  $v$  in the order: assign all unassigned  $u : d(v, u) \leq x$  to  $S_v$ .

**Claim 5.5.1.** For each  $(u, v) \in E$ , we have

$$\Pr[(u, v) \text{ cut by partition}] \leq \frac{O(\log n)}{r} \cdot d(u, v)$$

*Proof.* For  $e = (u, v)$ , denote  $d(z, e) = \min\{d(u, z), d(v, z)\}$ ,  $z \in V$ .

Order vertices in  $V$  such that  $d(z_1, e) \leq d(z_2, e) \leq \dots \leq d(z_n, e)$ .

First time one of  $u, v$  is assigned to some  $S_{z_i}$ , say  $z_i$  settles  $e = (u, v)$ . Furthermore, if exactly one of  $u, v$  assigned to  $S_{z_i}$ , say  $z_i$  cuts  $e$ .

$$\Pr[e \text{ cut by partition}] = \sum_{i=1}^n \Pr[e \text{ cut by } z_i]$$

Define  $a_i = d(z_i, e)$ ,  $b_i = \max\{d(u, z_i), d(v, z_i)\}$ . Then

$$\begin{aligned} \Pr[e \text{ cut by } z_i] &\leq \Pr[X \in [a_i, b_i) \text{ and } z_i \text{ comes before } z_1, \dots, z_{i-1} \text{ in the order}] \\ &= \Pr[X \in [a_i, b_i)] \cdot \Pr[z_i \text{ comes before } z_1, \dots, z_{i-1} \text{ in the order}] \\ &\leq \frac{b_i - a_i}{r/2} \cdot \frac{1}{i} \end{aligned}$$

Therefore,

$$\Pr[(u, v) \text{ cut by partition}] \leq \sum_{i=1}^n \frac{b_i - a_i}{r/2} \cdot \frac{1}{i} \leq \frac{d(u, v)}{r/2} O(\log n)$$

□

□

### 5.5.3 Max-cut

**Example 5.42** (Max-cut). Input: undirected graph  $G = (V, E)$ , edge weight  $\omega : E \rightarrow \mathbb{R}_{\geq 0}$ .

Our goal is to partition  $V$  into  $(A, B)$  to maximize  $\sum_{(i,j) \in E} \omega(i, j) \mathbf{1}[(i, j) \text{ cut by } (A, B)]$

$$\text{IP maximize } \sum_{(i,j) \in E} \omega(i, j) y_{ij}.$$

Subject to

$$x_i \in \{0, 1\}, \forall i \in V$$

$$y_{ij} \leq x_i + x_j$$

$$y_{ij} \leq 2 - x_i - x_j$$

We can relax to  $x_i \in [0, 1]$  to get an LP relaxation. However, LP=1 if  $x_i \equiv \frac{1}{2}, y_{ij} \equiv 1$ .

**Strengthened LP** Add  $y_{ij} + y_{jk} + y_{ki} \leq 2$ .

**Sherali-Adams** It is a way to construct LP relaxation, which try to describe the joint distribution of  $k$  variables.

For variables  $x_{I,\sigma}, I \subset [n], |I| \leq k, \sigma : I \rightarrow \{0, 1\}$ .

Then in this problem, objective is

$$\sum_{(i,j) \in E} \Pr[(i, j) \text{ is cut}] \sum_{(i,j) \in E} x_{\{i,j\},(0,1)} + x_{\{i,j\},(1,0)}$$

where  $x_{I,\sigma} \geq 0, \sum_{\sigma} x_{I,\sigma} = 1$ .

Consistency:

$$\sum_{\sigma' : J \setminus I \rightarrow \{0,1\}} x_{J,\sigma \cup \sigma'} = x_{I,\sigma}, \forall I \subset J, \sigma : I \rightarrow \{0, 1\}$$

**Theorem 5.43.** On dense graph (e.g. uniformly weighted,  $|E| \geq \frac{1}{100}|V|^2$ )

$SA(\frac{1}{\varepsilon}) + \text{rounding}$  achieves  $(1 - \varepsilon)$ -approximation.

**Theorem 5.44.**  $\forall \varepsilon > 0, \exists (\frac{1}{2} + \varepsilon)$ -Integrality Gap instance for even  $SA(n/100)$ .

So SA process is not good for max-cut.

$$\mathbf{IP}' \quad \max \sum_{(i,j) \in E} \omega_{ij} \frac{1 - x_i x_j}{2} \text{ subject to } x_i \in \{\pm 1\}, \forall i \in V$$

**Relaxation**  $x_i \rightsquigarrow \vec{v}_i \in \mathbb{R}^n$ .

**Semi-definite Programming relaxation**

$$\max \sum_{(i,j) \in E} \omega_{ij} \frac{1 - \langle \vec{v}_i, \vec{v}_j \rangle}{2}$$

subject to  $\|\vec{v}_i\|^2 = 1, \forall i \in V$ .

**Fact 5.45.**  $SDP \geq OPT$ .

**Definition 5.46** (SDP Standard Form).  $X \in \mathbb{R}^{n \times n}$ . Maximize  $\langle C, X \rangle = \text{Tr}(C^T X)$ ,  $X \in \mathbb{R}^{n \times n}$ , subject to  $\langle A_i, X \rangle = b_i, \forall i \in \{1, 2, \dots, m\}$  and  $X \geq 0$  positive semi-definite.

**Max-Cut SDP in matrix form** Maximize  $\langle \frac{1}{2}(D - A), X \rangle$  where  $D$  is degree matrix with  $D_{ii} = \sum_j a_{ij}$  and  $A$  adjacency matrix  $a_{ij} = \omega(i, j)/2$ , subject to  $\langle e_i e_i^T, X \rangle = 1, \forall i \in V$ .

**Separation Oracle** Use it to find solution and check it whether it is positive semi-definite by finding its min eigenvalue.

**Goemans-Williamson Rounding** [1995] Also called "hyperplane rounding".

1. Uniformly sample  $\vec{r} \sim S^{n-1}$

2.  $x_i = \text{sgn}(\langle \vec{r}, \vec{v}_i \rangle)$

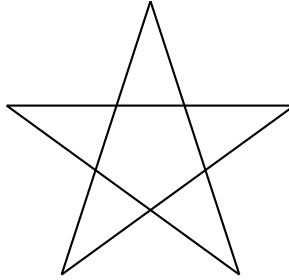
### Analysis

$$\begin{aligned} \mathbb{E}[\text{cut value}] &= \sum_{(i,j) \in E} \omega(i,j) \cdot \Pr[(i,j) \text{ cut}] \\ &= \sum_{(i,j) \in E} \omega_{ij} \cdot \frac{\arccos \langle v_i, v_j \rangle}{\pi} \\ &\geq \alpha_{GW} \sum_{(i,j) \in E} \omega_{ij} \cdot \frac{1 - \langle v_i, v_j \rangle}{2} \end{aligned}$$

where

$$\alpha_{GW} = \inf_{\rho \in [-1,1]} \frac{\frac{1}{\pi} \arccos \rho}{\frac{1}{2}(1 - \rho)} \approx 0.87856$$

**Integrality Gap** 5-Cycle:



Then  $\text{OPT} = 4/5$ ,

$$\text{SDP} \geq \frac{1 - \cos 144^\circ}{2} \approx 0.9045$$

Gap ratio  $\leq \frac{0.8}{0.9045} \approx 0.885$ .

**Embedded graph**  $G = (V, E, \omega)$ , consider  $V \subset S^{d-1} = \{\|x\|^2 = 1\}$ .

$$\text{Obj}(G) = \sum_{(\vec{x}, \vec{y}) \in E} \omega(\vec{x}, \vec{y}) \frac{1 - \langle \vec{x}, \vec{y} \rangle}{2}$$



**Fact 5.47.**  $\forall$  embedded graph  $G$ ,  $\text{SDP}(G) \geq \text{Obj}(G)$ .

**Gap Instance**  $G_d = (V, E)$ ,  $V = S^{d-1}$ ,  $E = V \times V$ .

Let  $\omega(\vec{x}, \vec{y})$  be the probability density of

$$\vec{x} \sim S^{d-1}, y \sim S^{d-1} \mid \langle \vec{x}, \vec{y} \rangle \leq \rho^*$$

Clearly, for this infinite graph,  $\text{SDP}(G) \geq \text{Obj}(G) \geq \frac{1-\rho^*}{2}$ .

For  $A \subset S^{d-1}$ ,

$$\mu_l(A) = \Pr_{\vec{x}, \vec{y} \sim S^{d-1}} [\vec{x} \in A \oplus \vec{y} \in A \mid \langle \vec{x}, \vec{y} \rangle \leq \rho]$$

**Theorem 5.48.** For  $\forall a \in [0, 1], \rho \in [-1, 1]$ ,  $\max_{A: \text{measure of } A = a} \{\mu_\rho(A)\}$  is achieved when  $A$  is a cap of  $S^d$ .

**Corollary 5.49.**  $\max\{\mu_{\rho^*}(A)\}$  is achieved when measure of  $A = \frac{1}{2}$ .

**Claim 5.5.2.**

$$\text{OPT}(G_d) \leq \frac{\arccos \rho^*}{\pi} + O\left(\frac{1}{\sqrt{d}}\right)$$

Leave as a homework.

Assume Unique Game Conjecture,  $\alpha_{GW}$  is the best Integrality Gap.

**Algorithmic Gap**

**Example 5.50** (Instance).  $H_d = (V, E, \omega)$ ,  $V = \{\pm \frac{1}{\sqrt{d}}\}^d$ .  $\omega \sim \text{distribution}$ ,  $\vec{x} \sim V$ ,

$$y_i = \begin{cases} x_i & \text{with prob } \frac{1+\rho^*}{2} \\ -x_i & \text{with prob } \frac{1-\rho^*}{2} \end{cases} \quad (5.1)$$

denoted as  $\vec{y} \sim_{\rho^*} \vec{x}$ .

Then  $\mathbb{E} x_i = \mathbb{E} y_i = 0$ ,  $\mathbb{E} x_i y_i = \left(\frac{1+\rho^*}{2}\right) \cdot 1 + \left(\frac{1-\rho^*}{2}\right) (-1) = \rho^*$ .

**Fact 5.51.**

$$\begin{aligned} \text{Obj}(H_d) &= \mathbb{E}_{(\vec{x}, \vec{y}) \sim \omega} \left[ \frac{1 - \langle \vec{x}, \vec{y} \rangle}{2} \right] = \frac{1}{2} - \frac{1}{2} \rho^* \\ \text{SDP} &\geq \frac{1}{2} - \frac{1}{2} \rho^* \end{aligned}$$

**Claim 5.5.3.**

$$\text{SDP} \leq \frac{1}{2} - \frac{1}{2} \rho^*$$

By the claim,

$$\begin{aligned} \mathbb{E}[\text{rounding}] &= \mathbb{E}_{(\vec{x}, \vec{y}) \sim \omega} \frac{\arccos \langle \vec{x}, \vec{y} \rangle}{\pi} \leq \frac{\arccos \rho^*}{\pi} + O\left(\frac{1}{\sqrt{d}}\right) \\ \text{OPT} &= \frac{1}{2} - \frac{1}{2} \rho^* \end{aligned}$$

**Fourier Analysis of Boolean Function** Boolean function:  $f : \{\pm 1\}^n \rightarrow \mathbb{R}$ .

Inner product  $\langle f, g \rangle = \mathbb{E}_{\vec{x} \sim \{\pm 1\}^n} [f(\vec{x}) \cdot g(\vec{x})]$

*Fourier basis*

$$\forall s \subset [n], \chi_s(\vec{x}) = \prod_{i \in s} x_i$$

$$\langle \chi_s, \chi_s \rangle = \mathbb{E}_{\vec{x}} \chi_s^2(\vec{x}) = 1.$$

$$\forall s \neq t, \langle \chi_s, \chi_t \rangle = \mathbb{E}_{\vec{x}} \chi_s(\vec{x}) \cdot \chi_t(\vec{x}) = \mathbb{E}_{\vec{x}} \chi_{s \Delta t}(\vec{x}) = \prod_{i \in s \Delta t} \mathbb{E}[x_i] = 0.$$

So  $\{\chi_s\}$  forms an orthonormal basis.

**Proposition 5.52** (Fourier expansion).  $\forall f : \{\pm 1\}^n \rightarrow \mathbb{R}$ ,

$$f = \sum_{s \subseteq [n]} \hat{f}(s) \cdot \chi_s$$

where  $\hat{f}(s) = \langle f, \chi_s \rangle$ .

**Theorem 5.53** (Parseval's).  $\forall f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ ,

$$\sum_{s \subseteq [n]} \hat{f}(s)^2 = \langle f, f \rangle = 1$$

**Noise Stability**  $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ .

$$\text{NS}_\rho(f) = \Pr_{\vec{x}, \vec{y} \sim_\rho \vec{x}}[f(\vec{x}) = f(\vec{y})]$$

Recall the notation of  $\sim_\rho$  in (5.1).

$$\begin{aligned} \text{NS}_\rho(f) &= \Pr_{\vec{x}, \vec{y} \sim_\rho \vec{x}}[f(\vec{x}) = f(\vec{y})] \\ &= \mathbb{E}_{\vec{x}, \vec{y} \sim_\rho \vec{x}} \frac{1 + f(\vec{x})f(\vec{y})}{2} \\ &= \frac{1}{2} + \frac{1}{2} \mathbb{E}_{\vec{x}, \vec{y} \sim_\rho \vec{x}} f(\vec{x}) \cdot f(\vec{y}) \\ &= \mathbb{E}_{\vec{x}, \vec{y} \sim_\rho \vec{x}} \left( \sum_s \hat{f}(s) \chi_s(\vec{x}) \right) \left( \sum_t \hat{f}(t) \chi_t(\vec{y}) \right) \\ &= \sum_{s, t} \hat{f}(s) \hat{f}(t) \mathbb{E}_{\vec{x}, \vec{y} \sim_\rho \vec{x}} \chi_s(\vec{x}) \chi_t(\vec{y}) \\ &= \sum_{s, t} \hat{f}(s) \hat{f}(t) \mathbb{E}_{\vec{x}} \chi_s(\vec{x}) \chi_t(\vec{x}) \cdot \mathbb{E}_{\vec{u}} \chi_t(\vec{u}) \quad \text{where } \vec{y} = \vec{x} \cdot \vec{u} \\ &= \sum_{s, t} \hat{f}(s)^2 \cdot \rho^{|s|} \end{aligned}$$

So

$$\text{NS}_\rho(f) = \frac{1}{2} + \frac{1}{2} \sum_s \hat{f}(s)^2 \cdot \rho^{|s|}$$

**Claim 5.5.4.** When  $\rho \in [-1, 0]$ ,

$$\begin{aligned} S_\rho(f) &= \sum_s \hat{f}(s)^2 \cdot \rho^{|s|} \\ &= \rho \cdot \sum_s \hat{f}(s)^2 \\ &= \rho \cdot \mathbb{E}_{\vec{x}}[f(x)^2] \end{aligned}$$

*Proof of Claim 5.5.3.* Consider any SDP solution

$$\vec{f} : V \rightarrow \mathbb{R}^{m-1}$$

$$\begin{aligned} \text{val}(\vec{f}) &= \mathbb{E}_{\vec{x}, \vec{y} \sim \omega} \frac{1 - \langle \vec{f}(\vec{x}), \vec{f}(\vec{y}) \rangle}{2} \\ &= \mathbb{E}_{\vec{x}, \vec{y} \sim \omega} \left[ \frac{1}{2} - \frac{1}{2} \sum_{i=1}^m f_i(\vec{x}) f_i(\vec{y}) \right] \\ &= \frac{1}{2} - \frac{1}{2} \sum_{i=1}^m \mathbb{E}_{\vec{x}, \vec{y} \sim \omega} [f_i(\vec{x}) f_i(\vec{y})] \\ &= \frac{1}{2} - \frac{1}{2} \sum_{i=1}^m S_{\rho^*}(f_i) \\ &\leq \frac{1}{2} - \frac{\rho^*}{2} \sum_{i=1}^m \mathbb{E}_{\vec{x} \sim V} [f_i(\vec{x})^2] \\ &= \frac{1}{2} - \frac{\rho^*}{2} \mathbb{E}_{\vec{x} \sim V} \sum_{i=1}^m [f_i(\vec{x})^2] \\ &= \frac{1}{2} - \frac{1}{2} \rho^* \end{aligned}$$

□

## 5.6 3-Coloring

**Example 5.54** (3-Coloring). Input:  $G = (V, E)$ , color  $v \in V$  with 3 colors so that there are no monochromatic edges.

**Example 5.55** (Min-3-Coloring). For 3-colorable graph, find a coloring with min number of colors.

**Lemma 5.56.** *Let  $\delta$  be the degree of  $G$ , then coloring  $G$  with  $(\delta + 1)$  colors is easy.*

**Claim 5.6.1.** Coloring  $G$  with  $n$  colors is easy.

**Widgerson's 3 vs.  $O(\sqrt{n})$  coloring algorithm**

**Case 1**  $\exists v \in V, \deg(v) > \theta$ . Then its neighbors  $G(N(v))$  are 2-colorable.

So color  $\{v\} \cup N(v)$  with 3 new colors.

**Case 2**  $\delta \leq \theta$ : color  $G$  with  $(\theta + 1)$  colors.

Then total number of colors used

$$3 \cdot \frac{n}{\theta + 1} + \theta + 1 \stackrel{\theta = \sqrt{n}}{=} O(\sqrt{n})$$

**Lemma 5.57.** *Let  $G$  be 3-colorable with degree  $\delta$ , then can efficiently color  $G$  with  $\delta^{\frac{1}{3}} \cdot \text{poly}(\log n)$  colors.*

If the lemma is true, use the method in case 1, then the cost will be

$$O\left(\frac{n}{\theta}\right) + O(\theta^{\frac{1}{3}}) \stackrel{\theta = n^{3/4}}{=} O(n^{1/4})$$

**3-coloring SDP**  $\langle v_i, v_j \rangle = -\frac{1}{2}, \forall (i, j) \in E, \|v_i\|^2 = 1, \forall i \in V.$

**Extract a large independent set** One can efficiently find  $S \subset V$  such that  $S$  is an  $IS$  and

$$\Pr[|S| \geq \frac{n}{\delta^{1/3}} \cdot \text{poly}(\log n)] \geq \frac{1}{2}$$

**Alg**

1. Sample  $\vec{r} = (r_1, \dots, r_n) \sim N(0, 1)^m$ .
2. Let  $S = \{i \in V : \langle v_i, r \rangle \geq t \wedge \langle v_j, r \rangle < t, \forall (i, j) \in E\}$ .

Let

$$\alpha(t) = \Pr_{\vec{r}}[\langle v_i, \vec{r} \rangle \geq t]$$

$$\beta(t) = \Pr_{\vec{r}}[\langle \vec{r}, \vec{v}_j \rangle \geq t \mid \langle \vec{r}, \vec{v}_i \rangle \geq t]$$

where  $(i, j) \in E$ .

$$E|S| = \sum_{i \in V} \alpha(t) \cdot \Pr[\langle \vec{r}, \vec{v}_j \rangle < t, \forall (i, j) \in E \mid \langle \vec{r}, \vec{v}_i \rangle \geq t] \geq \sum_{i \in V} \alpha(t)(1 - \delta \cdot \beta(t))$$

where

$$\alpha(t) = \Pr[r_1 \geq t] = \Phi(t)$$

$$\beta(t) = \Pr[-\frac{1}{2}r_1 + \frac{\sqrt{3}}{2}r_2 \geq t \mid r_1 \geq t] \leq \Pr[\frac{\sqrt{3}}{2}r_2 \geq \frac{3}{2}t] = \Phi(\sqrt{3}t)$$

**Claim 5.6.2.** For  $t \geq 1$ ,  $\Phi(t) = \Theta(\frac{1}{t} \cdot e^{-t^2/2})$ .

Therefore,

$$\alpha(t) \geq \frac{c}{t} e^{-t^2/2}$$

$$\beta(t) \leq \frac{c'}{t} e^{-3t^2/2} \leq t^2 O(\alpha(t)^3)$$

$$\mathbb{E}|S| = n \cdot \frac{c}{t} e^{-t^2/2} (1 - \delta t^2 \cdot O(\alpha(t)^3))$$

## 5.7 Sparsest Cut

**Example 5.58** (Sparsest Cut). Input: undirected  $G = (V, E, c)$ ,  $c : E \rightarrow \mathbb{R}_{\geq 0}$ .

Undirected demand  $G_D = (V, E, D)$ ,  $D : F \rightarrow \mathbb{R}_{\geq 0}$ .

Goal: minimize sparsity  $\Phi(S) = \frac{c(S, \bar{S})}{D(S, \bar{S})}$ ,  $S \subset V$ .

**Uniform Sparsest cut**  $G_D$  complete graph,  $D \equiv 1$ .

$$\Phi_U(S) = \frac{c(S, \bar{S})}{|S| \cdot |\bar{S}|}$$

**Graphic expansion** Define  $\phi(S) = \frac{c(S, \bar{S})}{|S|}$ , when  $|S| \leq \frac{n}{2}$

we have

$$\frac{1}{n}\phi(S) = \frac{c(S, \bar{S})}{|S| \cdot n} \leq \Phi_U(S) \leq \frac{c(S, \bar{S})}{|S| \cdot \frac{n}{2}} = \frac{2}{n}\phi(S)$$

$$\phi(S) = \min_{|S| \cdot |\bar{S}| \leq \frac{n}{2}} \{c(S, \bar{S})\}$$

So those problems are related.

### 5.7.1 $O(\log n)$ -approximation for sparsest cut

Nowadays, SDP-based algorithm can achieve  $O(\sqrt{\log n})$

**cut metric** Fix  $(S, \bar{S})$ . Let

$$\begin{aligned} \tilde{x}_{ij} &= 1[(i, j) \text{ cut by } (S, \bar{S})] \\ &= |1[i \in S] - 1[j \in S]| \end{aligned}$$

$\{\tilde{x}_{ij}\}$  is a metric.

Let  $x_{ij} = \frac{\tilde{x}_{ij}}{D(S, \bar{S})}$ , then

1)  $\{\tilde{x}_{ij}\}$  is also a metric.

$$2) \sum_{(i,j) \in F} D(i,j)x_{ij} = 1.$$

Therefore,

$$\sum_{(i,j) \in E} c(i,j) \cdot x_{ij} = \Phi(S)$$

**LP relaxation** Minimize  $\sum_{(i,j) \in E} c(i,j)x_{ij}$ .

Subject to  $\sum_{(i,j) \in F} D(i,j)x_{ij} = 1$ .

where  $x_{ij} \geq 0$ ,  $x_{ij} + x_{jk} \geq x_{ik}$ ,  $\forall i, j, k \in V$ .

**Definition 5.59** (Metric Embedding).  $(V, d)$  embeds into  $l_p$  ( $p \geq 1$ ) with distortion  $\alpha \geq 1$ . If  $\exists \vec{f}: V \rightarrow \mathbb{R}^K$  such that

$$\forall i, j \in V, \|\vec{f}(i) - \vec{f}(j)\|_p \leq d(i, j) \leq \alpha \|\vec{f}(i) - \vec{f}(j)\|_p$$

**Lemma 5.60.** *If we can embed  $(V, \{x_{i,j}\})$  into  $l_1$  with distortion  $\alpha$ , then  $\exists(S, \bar{S})$ , s.t.  $\Phi(S) \leq \alpha \text{LP}$*

*Proof.* Assume  $\vec{f} \geq \vec{0}$  WLOG.

Let  $M = \max_{i, k \in [K]} \{f_k(i)\}$ .

"Threshold cut":  $S_{k,\theta} = \{i \in V : f_k(i) \geq \theta\}$ .

Consider  $k \sim [K]$ ,  $\theta \sim \text{Uniform}$

$$\begin{aligned} \Pr[(i, j) \text{ cut}] &= \frac{1}{K} \sum_{k=1}^K \Pr[\theta \in [f_k(i), f_k(j)]] \\ &= \frac{1}{K} \sum_{k=1}^K |f_k(i) - f_k(j)| \end{aligned}$$



$$\begin{aligned}
\mathbb{E}[c(S_{k,\theta}, \bar{S}_{k,\theta})] &= \sum_{(i,j) \in E} c(i,j) \cdot \Pr[(i,j) \text{ cut}] \\
&= \frac{1}{KM} \sum_{(i,j) \in E} \sum_{k=1}^K |f_k(i) - f_k(j)| \cdot c(i,j) \\
&= \frac{1}{KM} \sum_{(i,j) \in E} \|\vec{f}(i) - \vec{f}(j)\|_1 \cdot c(i,j) \\
&\leq \frac{1}{KM} \sum_{(i,j) \in E} x_{ij} c(i,j)
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}[D(S_{k,\theta}, \bar{S}_{k,\theta})] &= \frac{1}{KM} \sum_{(i,j) \in F} \|\vec{f}(i) - \vec{f}(j)\|_1 D(i,j) \\
&\geq \frac{1}{\alpha \cdot KM} \sum_{(i,j) \in F} x_{ij} D(i,j)
\end{aligned}$$

Then

$$\frac{\mathbb{E}[c(S_{k,\theta}, \bar{S}_{k,\theta})]}{\mathbb{E}[D(S_{k,\theta}, \bar{S}_{k,\theta})]} \leq \alpha \frac{\sum_{(i,j)} x_{ij} c(i,j)}{\sum_{(i,j)} x_{ij} D(i,j)} = \alpha \text{LP}$$

□

**Theorem 5.61** (Bourgain' 1985). *Any  $n$ -point metric embeds into  $l_1$  with distortion  $\alpha = O(\log n)$ .*

**Theorem 5.62** (LLR 1995). *Any  $n$ -point metric embeds into  $l_p$ ,  $p \in [1, +\infty]$  with  $\alpha = O(\log n)$  and  $O(\log^2 n)$  dimensions*

**Embed tree into  $l_1$**  Any tree  $T = (V, E, d)$  isometrically embeds into  $l_1$ .

By induction,  $n = 2$ , let  $f(v_1) = 0, f(v_2) = d(v_1, v_2)$ .

Induction step: If  $T = T' \cup \{z\}$ ,  $\vec{f}'$  embeds  $T'$  into  $l_1$  for all  $v$  in  $T'$ , where  $\omega = d(u, v), (u, v) \in E$ .

$$\vec{f}(v) = (\vec{f}(v), 0), \vec{f}(z) = (\vec{f}(u), \omega)$$

To prove this theorem, we want to reduce to the case of trees. For metric  $(V, d)$ , our goal:  $\exists$  distribution of trees  $\mathcal{D}$  such that

$$\mathbb{E}_{T \sim \mathcal{D}} [d_T(u, v)] \leq d(u, v) \leq \alpha \mathbb{E}_{T \sim \mathcal{D}} [d_T(u, v)]$$

**Theorem 5.63.** *For any  $(V, d)$ ,  $\exists$  distribution  $\mathcal{D}$  over trees, such that  $\alpha = O(\log n)$ , and*

- 1)  $\frac{1}{\alpha} \leq \mathbb{E}_{T \sim \mathcal{D}} [d_T(u, v)] \leq d(u, v), \forall u, v \in V.$
- 2)  $d(u, v) \leq d_T(u, v), \forall u, v, T \in \text{Supp}(\mathcal{D})$

**Theorem 5.64 (Low-Diameter Decomposition).** *For  $(V, d)$ , radius  $r$ ,  $\exists$  random partition  $\{S_v\}_{v \in V}$  such that*

- 1)  $\forall v \in V, u \in S_v, d(u, v) \leq r$
- 2) *For each  $(u, v)$  such that  $d(u, v) \leq r/4$ ,*

$$\Pr[(u, v) \text{ cut}] \leq \frac{d(u, v)}{r} \cdot O\left(\log \frac{|B(u, 2r)|}{|B(u, r/4)|}\right)$$

*Proof.* **Generating the Partition**

- 1) Sample  $X \sim \text{Unif}[r/2, r]$
- 2) Randomly order  $\pi$  vertices in  $V$
- 3) For each  $v \in V$  in the order  $\pi$ : let  $S_v = \{u : d(u, v) \leq X \text{ and } u \text{ not yet assigned}\}.$

**Definition 5.65.**  $d(z, (u, v)) = \min(d(z, u), d(z, v))$

Fix  $(u, v)$ , order vertices in  $V$ ,  $(z_1, \dots, z_n)$  such that

$$d(z_1, (u, v)) \leq d(z_2, (u, v)) \leq \dots \leq d(z_n, (u, v))$$

$z_i$  settles  $(u, v)$  if  $z_i$  is the first partition in  $\pi$  s.t.  $u \in S_{z_i}$  or  $v \in S_{z_i}$ .

$z_i$  cuts  $(u, v)$  if  $z_i$  settles  $(u, v)$  and exactly one of  $u, v$  in  $S_{z_i}$ .

$$\Pr[(u, v) \text{ cut}] = \sum_{i=1}^n \Pr[z_i \text{ cuts } (u, v)]$$

$$\begin{aligned} \Pr[z_i \text{ cuts } (u, v)] &\leq \Pr[z_i \text{ comes before } z_1, \dots, z_{i-1} \text{ in } \pi \text{ and } X \in [d(z_i, (u, v)), \max(d(z_i, u), \\ &= \frac{1}{i} \Pr[X \in [a_i, b_i)] \end{aligned} \quad (5.2)$$

1.  $a_i \leq \frac{r}{4}$ : we have  $b_i \leq \frac{r}{4} + \frac{r}{4} = \frac{r}{2}$ . So (5.2) = 0

2.  $a_i > r$ . Then (5.2) = 0

3.  $a_i \in (r/4, r]$ . Then (5.2)  $\leq \frac{d(u, v)}{r/2}$

So

$$\begin{aligned} \Pr[z_i \text{ cuts } (u, v)] &\leq \frac{2d(u, v)}{r} \cdot \frac{1}{i} \cdot 1[a_i \in (\frac{r}{4}, r]] \\ \Pr[(u, v) \text{ cuts}] &\leq \frac{2d(u, v)}{r} \cdot \sum_{i=1}^n \frac{1}{i} 1[a_i \in (\frac{r}{4}, r]] \leq O\left(\frac{d(u, v)}{r}\right) \cdot \ln \frac{\beta}{\alpha} \end{aligned}$$

□

TreeEmbed( $U, \beta$ ) where  $2^\beta \geq \max_{u, v \in U} d(u, v)$ .

1. IF  $|U| = 1$  THEN return the single node in  $U$

2. INVOKE LDD with  $r = 2^{\beta-2}$ , obtained  $\{S_u\}_{u \in U}$ .

3. FOR each nonempty  $S_u$ , invoke TreeEmbed( $S_u, \beta - 1$ )

4. RETURN a tree that  $U$  connects to  $T_{u_i}$  for each  $u_i$

**Claim 5.7.1.**  $\forall u, v \in U, d(u, v) \leq d_T(u, v)$

**Expansion** If  $G = (V, E)$   $d$ -regular.

$$\phi(S) = \Pr_{v \in S, u \in N(G)}[u \notin S] = \frac{|\text{edges}(S, \bar{S})|}{d \cdot |S|}$$

$$\phi(G) = \min_{S: 1 \leq |S| \leq \frac{n}{2}} \{\phi(S)\}$$

**Cheager's Inequality**

$$\frac{1}{2} \lambda_2(L_G) \leq \phi(G) \leq \sqrt{2 \lambda_2(L_G)}$$

where **Laplacian**  $L_G = I - \frac{1}{d}A$ .

**Spectral Graph**  $x \in \vec{\mathbb{R}}^v$ ,

$$\vec{x}^T L_G \vec{x} = \frac{1}{d} \sum_{(u,v) \in E} (x_u - x_v)^2$$

**Claim 5.7.2.**

1.  $L_G \geq 0$
2.  $\lambda_{\min}(L_G) = 0$
3. Consider  $G$  disconnected, then  $\lambda_2(L_G) = 0$ .

4.

$$\lambda_2(L_G) = \min_{\vec{x} \perp \vec{1}} \frac{\vec{x}^T L_G \vec{x}}{\vec{x}^T \vec{x}}$$

*Proof of  $\frac{1}{\lambda_2} \leq \phi(G)$ .*

Consider  $|S^*| \leq \frac{n}{2}$ ,  $\phi(S^*) = \phi(G)$ .

Our goal is to construct  $\vec{x} : \vec{x} \perp \vec{1}$  and  $\frac{\vec{x}^T L_G \vec{x}}{\vec{x}^T \vec{x}} \leq 2\phi(S^*)$ .

$\forall i \in V$ , let

$$x_i = \begin{cases} \alpha & i \in S^* \\ \beta & i \notin S^* \end{cases}$$

where  $\alpha \cdot |S^*| + \beta \cdot (n - |S^*|) = 0$ .

So we let  $\alpha = -\frac{1}{|S^*|}$ ,  $\beta = \frac{1}{n - |S^*|}$ .

$$\vec{x}^T \vec{x} = |S^*| \cdot \frac{1}{|S^*|^2} + (n - |S^*|) \cdot \frac{1}{(n - |S^*|)^2} = \frac{n}{|S^*|(n - |S^*|)}$$

$$\begin{aligned} \vec{x}^T L_G \vec{x} &= \sum_{(i,j) \in E} (x_i - x_j)^2 \\ &= |\text{edge}(S^*, \bar{S}^*)| \cdot \left( \frac{1}{|S^*|} + \frac{1}{n - |S^*|} \right)^2 \end{aligned}$$

Then

$$\frac{\vec{x}^T L_G \vec{x}}{\vec{x}^T \vec{x}} = |\text{edge}(S^*, \bar{S}^*)| \cdot \frac{n}{|S^*|(n - |S^*|)} \leq \phi(S^*)$$

□

*Proof of  $\phi(G) \leq \sqrt{2\lambda_2(L_G)}$ .*

Let  $\vec{x} \perp \vec{1}$  such that  $\frac{\vec{x}^T L_G \vec{x}}{\vec{x}^T \vec{x}} = \lambda_2$

Our goal is to construct  $S : |S| \leq \frac{n}{2}$  such that  $\phi(S) \leq \sqrt{2\lambda_2}$ .

Consider  $\vec{x}^+$  where  $(\vec{x}^+)_i = \max\{x_i, 0\}$ .

Assume WLOG,  $|\text{Supp}(\vec{x}^+)| \leq n/2$ .

**Claim 5.7.3.**

$$\frac{(\vec{x}^+)^T L_G \vec{x}^+}{(\vec{x}^T \vec{x})} \leq \lambda_2$$

*Proof.* For each  $i \in \text{Supp}(\vec{x}^+)$ .

$$\begin{aligned} (L_G \vec{x}^+)_i &= (\vec{x}^+)_i - \frac{1}{d} \sum_{j \sim i} (\vec{x}^+)_j \\ &\leq x_i - \frac{1}{d} \sum_{j \sim i} x_j \\ &= (L_G \vec{x})_j = \lambda_2 x_j = \lambda (\vec{x}^+)_j \end{aligned}$$

$$\begin{aligned} (\vec{x}^+)^T L_G (\vec{x}^+) &= \sum_{i \in \text{Supp}(\vec{x}^+)} (\vec{x}^+)_i (L_G \vec{x}^+)_i \\ &\leq \sum_{i \in \text{Supp}(\vec{x}^+)} (\vec{x}^+)_i \lambda_2 (\vec{x}^+)_i \\ &= \lambda_2 (\vec{x}^+)^T (\vec{x}^+) \end{aligned}$$

□

Let  $\vec{y} = \vec{x}^+$ . Assume WLOG,  $\max_i y_i = 1$ .

We already have

$$\frac{\vec{y}^T L_G \vec{y}}{\vec{y}^T \vec{y}} \leq \lambda_2 \text{ and } \text{Supp}(\vec{y}) \leq \frac{n}{2}$$

Let  $S_t = \{i : y_i^2 \geq t\}$ .

**Claim 5.7.4.**  $\exists t \in (0, 1] : \phi(S_t) \leq \sqrt{2\lambda_2}$

*Proof.* Choose  $t$  uniformly from  $(0, 1)$ .

$$\begin{aligned}
\mathbb{E} |\text{edges}(S_t, \bar{S}_t)| &= \sum_{(i,j) \in E} \Pr[i, j \text{ cut}] \\
&= \sum_{(i,j) \in E} |y_i^2 - y_j^2| = \sum_{(i,j) \in E} (y_i + y_j) \cdot |y_i - y_j| \\
&\leq \sqrt{\sum_{(i,j) \in E} (y_i + y_j)^2} \cdot \sqrt{\sum_{(i,j) \in E} (y_i - y_j)^2} \\
&\leq \sqrt{2d} \sum_{i \in V} y_i^2 \cdot \sqrt{d \cdot \vec{y}^T L_G \vec{y}}
\end{aligned}$$

$$\mathbb{E}[d|S_t|] = d \sum_{i \in V} \Pr[i \in S_t] = d \sum_{i \in V} y_i^2$$

$$\frac{\mathbb{E} |\text{edges}(S_t, \bar{S}_t)|}{\mathbb{E}[d|S_t|]} \leq \sqrt{\frac{2\vec{y}^T L_G \vec{y}}{\sum_{i \in V} y_i^2}} \leq \sqrt{2\lambda_2}$$

□

□

## 5.8 Hardness of Approximating MAXE3LIN

**Theorem 5.66.**  $\forall \delta > 0$ , MAXE3LIN $_{1-\delta, \frac{1}{2}+\delta}$  is NP-Hard.

We will use "long code" to encode instances of Label-Cover problem.

$$f_u : \{\pm 1\}^K \rightarrow \{\pm 1\}, f_u(\vec{x}) = x_{\sigma(u)}$$

**Linear Test:** A function  $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$  is linear if  $f(\vec{x}) \cdot f(\vec{y}) = f(\vec{x}\vec{y}), \forall \vec{x}, \vec{y} \in \{\pm 1\}^n$ .

**BLR Test:** Uniformly Randomly sample  $\vec{x}, \vec{y} \sim \{\pm 1\}^n$ . Pass if  $f(\vec{x})f(\vec{y})f(\vec{x}\vec{y}) = 1$ .

Completeness is trivial.

Soundness: If  $f$  is  $\delta$ -far from all linear functions, then  $\Pr[\text{Pass}] \leq 1 - \delta$ .

*Proof.*

$$\begin{aligned}
\Pr[\text{pass}] &= \frac{1}{2} + \frac{1}{2} \mathbb{E}_{\vec{x}, \vec{y}} [f(\vec{x})f(\vec{y})f(\vec{x}\vec{y})] \\
&= \frac{1}{2} + \frac{1}{2} \mathbb{E}_{\vec{x}, \vec{y}} \left[ \left( \sum_S \hat{f}(S) \chi_S(\vec{x}) \right) \left( \sum_T \hat{f}(T) \chi_T(\vec{y}) \right) \left( \sum_U \hat{f}(U) \chi_U(\vec{x}\vec{y}) \right) \right] \\
&= \frac{1}{2} + \frac{1}{2} \sum_{S, T, U} \hat{f}(S) \hat{f}(T) \hat{f}(U) \left( \mathbb{E}_{\vec{x}} \chi_S(\vec{x}) \chi_U(\vec{x}) \right) \left( \mathbb{E}_{\vec{y}} \chi_T(\vec{y}) \chi_U(\vec{y}) \right) \\
&= \frac{1}{2} + \frac{1}{2} \sum_S \hat{f}(S)^3
\end{aligned}$$

$$\text{So } \Pr[\text{pass}] = \frac{1}{2} + \frac{1}{2} \sum_S \hat{f}(S)^3 \leq \frac{1}{2} + \frac{1}{2} \left( \sum_S \hat{f}(S)^2 \right) \max_S \hat{f}(S) = \frac{1}{2} + \frac{1}{2} \max_S \hat{f}(S).$$

Then in all  $1 - \delta < \frac{1}{2} + \frac{1}{2} \max_S \{\hat{f}(S)\}$  i.e.  $\exists S^*, \hat{f}(S^*) > 1 - 2\delta$ .

$$\Pr_{\vec{x}} [f(\vec{x}) \neq \chi_{S^*}(\vec{x})] = \frac{1}{2} - \frac{1}{2} \mathbb{E}_{\vec{x}} [f(\vec{x}) \chi_{S^*}(\vec{x})] = \frac{1}{2} - \frac{1}{2} \hat{f}(S^*) < \delta$$

□

**Dictatorship Test** Our dictator function is  $\{f(\vec{x}) = x_i\}_{i \in \{1, 2, \dots, n\}}$

1. Choose  $\vec{x}, \vec{y} \sim \{\pm 1\}^m$

2. Choose  $\vec{\mu} \in \{\pm 1\}^n, \mu_i = \begin{cases} 1 & \text{w.p. } 1 - \varepsilon \\ -1 & \text{w.p. } \varepsilon \end{cases}.$

3. Accept if  $f(\vec{x})f(\vec{y})f(\vec{x}\vec{y}\vec{\mu}) = 1$ .

Completeness: If  $f(\vec{x}) = x_i$ , then  $\Pr[\text{pass}] = 1 - \varepsilon$ .



Soundness:

$$\begin{aligned}
\Pr[\text{pass}] &= \frac{1}{2} + \frac{1}{2} \mathbb{E}_{\vec{x}, \vec{y}, \vec{\mu}} [f(\vec{x})f(\vec{y})f(\vec{x}\vec{y}\vec{\mu})] \\
&= \frac{1}{2} + \frac{1}{2} \mathbb{E}_{\vec{x}, \vec{y}, \vec{\mu}} \left( \sum_S \hat{f}(S) \chi_S(\vec{x}) \right) \left( \sum_T \hat{f}(T) \chi_T(\vec{y}) \right) \left( \sum_U \hat{f}(U) \chi_U(\vec{x}\vec{y}\vec{\mu}) \right) \\
&= \frac{1}{2} + \frac{1}{2} \sum_{S, T, U} \hat{f}(S) \hat{f}(T) \hat{f}(U) \left( \mathbb{E}_{\vec{x}} \chi_{S \Delta U}(\vec{x}) \right) \left( \mathbb{E}_{\vec{y}} \chi_{T \Delta U}(\vec{y}) \right) \left( \mathbb{E}_{\vec{\mu}} \chi_U(\vec{\mu}) \right) \\
&= \frac{1}{2} + \frac{1}{2} \sum_S \hat{f}(S)^3 (1 - 2\varepsilon)^{|S|}
\end{aligned}$$

If  $\Pr[\text{pass}] \geq \frac{1}{2} + \delta$ , then

$$2\delta \leq \sum_S \hat{f}(S)^3 (1 - 2\varepsilon)^{|S|} \leq \max_S \{ \hat{f}(S) (1 - 2\varepsilon)^{|S|} \} \sum_S \hat{f}(S)^2$$

Therefore,  $\exists S \subset [n]$  s.t.  $\hat{f}(S)(1 - 2\varepsilon)^{|S|} \geq 2\delta$ . i.e.  $\begin{cases} \hat{f}(S) \geq 2\delta \\ |S| \leq \frac{\log 2\delta}{\log(1-\varepsilon)} = O(\frac{1}{\varepsilon} \log \frac{1}{\delta}) \end{cases}$

So we prove that if  $\Pr[\text{pass}] \geq \frac{1}{2} + \delta$ , then can "list decode"  $f$  into  $O(\frac{1}{\varepsilon \delta^2} \log \frac{1}{\delta})$

Now back to the problem. For a label-cover game  $(K, L)$ , try to reduce to MaxE3LIN problem.

Proof:  $f_u : \{\pm 1\}^K \rightarrow \{\pm 1\}$ ,  $f_v : \{\pm 1\}^L \rightarrow \{\pm 1\}$ .

Verifier:

1. Randomly pick  $e = (u, v) \in E$

2. Sample  $\vec{x} \sim \{\pm 1\}^K$ ,  $\vec{y} \sim \{\pm 1\}^L$ ,  $b \sim \{\pm 1\}$ ,  $\vec{\mu} \sim \{\pm 1\}^L$ ,  $\mu_i = \begin{cases} 1 & \text{w.p. } 1 - \varepsilon \\ -1 & \text{w.p. } \varepsilon \end{cases}$

3. Test  $f_u(\vec{x})g_v(\vec{y})g_v((\vec{x} \circ \pi_e)\vec{y}\vec{\mu} \cdot b) = b$ , where

$$(\vec{x} \circ \pi_e)_i = x_{\pi_e(i)}$$

Random bit  $b$  is for not to be hacked.

Completeness: If  $\sigma$  satisfies all  $LC$  constraints. Let  $f_u(\vec{x}) = x_{\sigma(u)}$ ,  $g_v(\vec{y}) = y_{\sigma(v)}$ .

Then

$$\begin{aligned} f_u(\vec{x})g_v(\vec{y})g_v((\vec{x} \circ \pi_e)\vec{y}\vec{\mu} \cdot b) &= x_{\sigma(u)}y_{\sigma(v)}x_{\pi_e(\sigma(v))}y_{\sigma(v)}\mu_{\sigma(v)} \cdot b \\ &= \mu_{\sigma(v)} \cdot b \end{aligned}$$

Therefore,  $\Pr[\text{accept}] = 1 - \varepsilon$ .

Soundness: If  $\Pr[\text{accept}] > \frac{1}{2} + \delta$ , then we prove that  $\exists$  LC solution value  $\geq \eta$ .

Fix  $e = (u, v) \in E$ ,

$\Pr[e \text{ passes test}]$

$$\begin{aligned} &= \frac{1}{2} + \frac{1}{2} \mathbb{E}_{\vec{x}, \vec{y}, \vec{\mu}, b} [bf_u(\vec{x})g_v(\vec{y})g_v((\vec{x} \circ \pi_e)\vec{y}\vec{\mu} \cdot b)] \\ &= \frac{1}{2} + \frac{1}{2} \mathbb{E} \left[ \left( \sum_S \hat{f}_u(S) \chi_S(\vec{x}) \right) \left( \sum_T \hat{g}_v(T) \chi_T(\vec{y}) \right) \left( \sum_U \hat{g}_v(U) \chi_U((\vec{x} \circ \pi_e) \cdot \vec{y} \cdot \vec{\mu} \cdot b) \right) \cdot b \right] \\ &= \frac{1}{2} + \frac{1}{2} \sum_{S, T, U} \hat{f}_u(S) \hat{g}_v(T) \hat{g}_v(U) \mathbb{E} [\chi_S(\vec{x}) \chi_T(\vec{y}) \chi_U((\vec{x} \circ \pi_e) \vec{y} \vec{\mu} b)] \\ &= \frac{1}{2} + \frac{1}{2} \sum_{S, T, U} \hat{f}_u(S) \hat{g}_v(T) \hat{g}_v(U) \mathbb{E}_{\vec{x}, b} [\chi_S(\vec{x}) \chi_U((\vec{x} \circ \pi_e) \cdot b) \cdot b] \\ &\quad \mathbb{E}_{\vec{y}} [\chi_T(\vec{y}) \chi_U(\vec{y})] \mathbb{E}_{\vec{\mu}} [\chi_U(\vec{\mu})] \\ &= \frac{1}{2} + \frac{1}{2} \sum_{S, T} \hat{f}_u(S) \hat{g}_v(T)^2 (1 - 2\varepsilon)^{|T|} \mathbb{E}_{\vec{x}, b} [\chi_S(\vec{x}) \chi_U((\vec{x} \circ \pi_e) \cdot b) \cdot b] \\ &= \frac{1}{2} + \frac{1}{2} \sum_{S, T} \hat{f}_u(S) \hat{g}_v(T)^2 (1 - 2\varepsilon)^{|T|} \mathbb{E}_{\vec{x}, b} [\chi_S(\vec{x}) \chi_T(\vec{x} \circ \pi_e) \cdot b^{|T|+1}] \\ &= \frac{1}{2} + \frac{1}{2} \sum_{S, T} \hat{f}_u(S) \hat{g}_v(T)^2 (1 - 2\varepsilon)^{|T|} \mathbf{1}[|T| \text{ odd}] \cdot \mathbb{E}_{\vec{x}} [\chi_S(\vec{x}) \chi_T(\vec{x} \circ \pi_e)] \\ &= \frac{1}{2} + \frac{1}{2} \sum_{S, T} \hat{f}_u(S) \hat{g}_v(T)^2 (1 - 2\varepsilon)^{|T|} \mathbf{1}[|T| \text{ odd}] \cdot \mathbb{E}_{\vec{x}} \left[ \prod_{i \in S} x_i \prod_{j \in T} x_{\pi_e(j)} \right] \\ &= \frac{1}{2} + \frac{1}{2} \sum_{S, T} \hat{f}_u(S) \hat{g}_v(T)^2 (1 - 2\varepsilon)^{|T|} \mathbf{1}[|T| \text{ odd}] \cdot \underbrace{\mathbb{E}_{\vec{x}} \left[ \prod_{i \in [K]} x_i^{1[|i \in S| + |\pi_e^{-1}(i) \cap T|]} \right]}_{(*)} \end{aligned}$$

Let  $\pi_e^{\text{odd}}(T) = \{i \in [K] : |\pi_e^{-1}(i) \cap T| \text{ odd}\}$ . Then

$$\begin{aligned} (*) \neq 0 &\Leftrightarrow \forall i \in [K], [i \in S] + |\pi_e^{-1}(i) \cap T| \text{ even} \\ &\Leftrightarrow \pi_e^{\text{odd}}(T) = S. \end{aligned}$$

Therefore,

$$\Pr[e \text{ pass}] = \frac{1}{2} + \underbrace{\frac{1}{2} \sum_{|T|:\text{odd}} \hat{f}_u(\pi_e^{\text{odd}}(T)) \hat{g}_v(T)^2 (1 - 2\varepsilon)^{|T|}}_{(**)}$$

If  $\frac{1}{2} + \gamma \leq \Pr[\text{pass}]$ , then  $(**) \geq 2\gamma$ .

Since  $\sum_T \hat{g}_v(T)^2 = 1$ ,

$$\begin{aligned} 2\gamma &\leq \mathbb{E}_{T \sim \hat{g}_v^2} \left[ \mathbf{1}[|T| \text{ odd}] \hat{f}_u(\pi_e^{\text{odd}}(T)) (1 - 2\varepsilon)^{|T|} \right] \\ &\Rightarrow \Pr_{T \sim \hat{g}_v^2} \left[ \mathbf{1}[|T| \text{ odd}] \hat{f}_u(\pi_e^{\text{odd}}(T)) (1 - 2\varepsilon)^{|T|} \geq \gamma \right] \geq \gamma \end{aligned}$$

Call  $T$  good if  $T$  satisfies the above condition.

If  $T$  is good, we have

- 1)  $|T| \text{ odd}$ , then  $\pi_e^{\text{odd}}(T) \neq \emptyset, \forall \pi$ .
- 2)  $(1 - 2\varepsilon)^{|T|} \geq \gamma \Rightarrow |T| \leq O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ , denote  $B = O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$
- 3)  $\hat{f}_v(\pi_e^{\text{odd}}(|T|))^2 \geq \gamma^2$

**Definition 5.67.**  $\forall u \in U$ ,  $\text{Sugg}(u) = \{j \in T \mid \exists S : \hat{f}_u(S)^2 \geq \gamma^2, |U| \leq B\}$ .

Then  $|\text{Sugg}(u)| \leq O(\frac{1}{r^2} \cdot \frac{1}{\varepsilon} \cdot \log \frac{1}{\varepsilon})$ .

Sample  $\text{Sugg}(v) = T \sim \hat{g}_v^2$ .

Decoding  $\sigma(u) \sim \text{Unif}(\text{Sugg}(u)), \sigma(v) \sim \text{Unif}(\text{Sugg}(v))$ .

$$\begin{aligned}
\Pr[\sigma \text{ satisfies } e = (u, v)] &= \Pr[\sigma(u) = \pi_e(\sigma(v))] \\
&= \Pr_{\substack{T \sim \hat{g}_v^2, \beta \sim T \\ \alpha \sim \text{Sugg}(u)}} [\alpha = \pi_e(\beta)] \\
&= \Pr[T \text{ good}] \cdot \Pr_{\substack{\beta \sim T \\ \alpha \sim \text{Sugg}(u)}} [\alpha = \pi_e(\beta), T \text{ good}] \\
&\geq \gamma \cdot \frac{1}{B} \cdot \frac{1}{|\text{Sugg}(u)|} \\
&\geq \Omega(\gamma^3 \varepsilon^2 \log^{-2} \frac{1}{\gamma})
\end{aligned}$$

$$\mathbb{E}_e[\gamma(e)] \geq \delta \Rightarrow \Pr_e(e \text{ satisfies } \pi_e) \geq \Omega(\delta^3 \varepsilon^2 / \log^2 \frac{1}{\delta}) = \eta$$

Here we finally prove that  $\text{E3LIN}_{1-\varepsilon, 1/2+\delta}$  is NP-Hard.

**Corollary 5.68.**  $\forall \varepsilon > 0$ ,  $\text{Max3SAT}_{1-\varepsilon, \frac{7}{8}+\varepsilon}$  is NP-Hard.

*Proof.* Reduce the instance of 3LIN to 3SAT as follows:

$$\begin{aligned}
&3\text{LIN}(I) \rightarrow 3\text{SAT}(J) \\
x_i + x_j + x_k \equiv 0 &\rightarrow \begin{cases} \bar{x}_i \vee \bar{x}_j \vee \bar{x}_k \\ \bar{x}_i \vee x_j \vee x_k \\ x_i \vee \bar{x}_j \vee x_k \\ x_i \vee x_j \vee \bar{x}_k \end{cases}
\end{aligned}$$

$$x_i + x_j + x_k \equiv 1 \rightarrow \begin{cases} x_i \vee x_j \vee x_k \\ x_i \vee \bar{x}_j \vee \bar{x}_k \\ \bar{x}_i \vee x_j \vee \bar{x}_k \\ \bar{x}_i \vee \bar{x}_j \vee x_k \end{cases}$$

Then  $\text{OPT}(I) \geq 1 - \varepsilon \Rightarrow \text{OPT}(J) \geq 1 - \varepsilon$ .

$\text{OPT}(I) < 1/2 + \varepsilon \Rightarrow \text{OPT}(J) < 7/8 + \varepsilon$

□

## 5.9 Hardness of Max-Cut

**Corollary 5.69.**  $\forall \varepsilon > 0$ ,  $\left(\frac{16}{17} + \varepsilon\right)$ -approximation Max-Cut is NP-Hard.

**Remark 5.70.** Assume **Unique Games Conjecture**,  $\alpha_{GW} + \varepsilon$ -approximation Max-Cut  $\notin \text{P}$ .

**Definition 5.71.** **Unique Label Cover Game** is the label cover game  $(K, K)$  that  $\pi_{uv}$  is a permutation over  $[K]$ .

**Fact 5.72.**  $\text{ULC} \in \text{P}$

**Conjecture 5.73.**  $\forall \delta > 0$ ,  $\exists K > 0$  such that  $\text{ULC}_{1-\delta, \delta} \notin \text{P}$

*Proof of Remark 5.70.*

**Cut Dictatorship Test:**  $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ .

Cut test:  $f(\vec{x}) \neq f(\vec{y})$ .

Parameter:  $\rho$ .

1. Uniformly sample  $\vec{x} \sim \{\pm 1\}^n$

2. Sample  $\vec{\mu} \in \{\pm 1\}^n$

$$\mu_i = \begin{cases} 1 & \text{w.p. } \frac{1+\rho}{2} \\ -1 & \text{w.p. } \frac{1-\rho}{2} \end{cases} \quad (\mathbb{E} \mu_i = \rho)$$

Test  $f(\vec{x}) \neq f(\vec{x}\vec{\mu})$

$$\begin{aligned} \Pr[f \text{ pass}] &= \mathbb{E}_{\vec{x}, \vec{\mu}} \left[ \frac{1 - f(\vec{x})f(\vec{x}\vec{\mu})}{2} \right] \\ &= \frac{1}{2} - \frac{1}{2} \mathbb{E}_{\vec{x}, \vec{\mu}} \left( \sum_S \hat{f}(S) \chi_S(\vec{x}) \right) \left( \sum_T \hat{f}(T) \chi_T(\vec{x}\vec{\mu}) \right) \\ &= \frac{1}{2} - \frac{1}{2} \sum_{S,T} \hat{f}(S) \hat{f}(T) \left( \sum_{\vec{x}} \chi_S(\vec{x}) \chi_T(\vec{x}) \right) \left( \sum_{\vec{\mu}} \chi_T(\vec{\mu}) \right) \\ &= \frac{1}{2} - \frac{1}{2} \sum_S \hat{f}(S)^2 \rho^{|S|} \\ &= \frac{1}{2} - \frac{1}{2} \mathbb{S}_\rho(f) \end{aligned}$$

We investigate some major functions:

1. Dictators:  $f(\vec{x}) = x_i$

$$\Pr[f \text{ pass}] = \frac{1}{2} - \frac{1}{2} \rho$$

2. Constant functions:  $f \equiv 0, f \equiv 1$ .

$$\Pr[f \text{ pass}] = 0$$

3. Parity functions:  $f = \chi_S(\vec{x})$ .

$$\Pr[f \text{ pass}] = \frac{1}{2} - \frac{1}{2} \rho^{|S|}$$

4. Majority function:  $f(\vec{x}) = \text{sgn}\left(\frac{x_1 + \dots + x_n}{\sqrt{n}}\right)$

$$\begin{pmatrix} \frac{x_1 + \dots + x_n}{\sqrt{n}} \\ \frac{\sum x_i \mu_i}{\sqrt{n}} \end{pmatrix} \rightarrow \mathcal{N}(0, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix})$$

$$\Pr[f \text{ passes}] \approx \Pr_{\begin{pmatrix} g_1 \\ g_2 \end{pmatrix} \sim \mathcal{N}(0, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix})} [\text{sgn}(g_1) \neq \text{sgn}(g_2)] = \frac{\arccos \rho}{\pi}$$

5. Linear Threshold function (LFT)

$$f(\vec{x}) = \text{sgn}(\vec{a}^T \vec{x}), \|\vec{a}\|_2 = 1$$

Then Majority function is a special case  $\vec{a} = (1/\sqrt{n}, \dots, 1/\sqrt{n})$  of LFT.

Dictator function is also a special case  $\vec{a} = (0, 0, \dots, 0, 1, 0, \dots, 0)$  of LFT.

**Theorem 5.74** (High-Dim Berry Esseen). *Let  $\vec{Y}_1, \dots, \vec{Y}_n$  be independent random variables in  $\mathbb{R}^d$ , where  $\mathbb{E} Y_i = \vec{0}$ ,  $\sum_{i=1}^n \text{Val}[Y_i] = I_d$ . Let  $\vec{Y} = \vec{Y}_1 + \dots + \vec{Y}_n$ . For all measurable convex sets  $A \subset \mathbb{R}^d$ , we have*

$$\left| \Pr[\vec{Y} \in A] - \Pr_{\vec{g} \sim N(0, I_d)}[\vec{g} \in A] \right| \leq (42d^{1/4} + 16) \sum_{i=1}^n \mathbb{E} \|Y_i\|_2^3$$

**Lemma 5.75.** *If  $f = \text{sgn}(\vec{a}^T \vec{x})$ ,  $\|\vec{a}\|_2 = 1$ . Then*

$$\Pr[f \text{ passes}] = \frac{\arccos \rho}{\pi} \pm \frac{O(1)}{\sqrt{(1 - \rho^2)^3}} \max_i |a_i|$$

Completeness: If  $f$  is a dictator function, then  $\Pr[\text{pass}] = \frac{1 - \rho}{2}$ .

Soundness:  $f$  far from dictator, then  $\Pr[\text{textpass}] \sim \frac{\arccos \rho}{\pi} \Leftrightarrow \Pr[\text{pass}] >$

$\frac{\arccos \rho}{\pi} + \varepsilon$ , then  $f$  has a few "influential dimensions" which can be decoded for

ULC.

**Definition 5.76 (Influence).** Given  $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ . For any  $i \in [n]$ , define

$$\begin{aligned}
\text{Inf}_i(f) &= \Pr_{\vec{x} \sim \{\pm 1\}^n} [f(\vec{x}) \neq f(\vec{x}^{\otimes i})] \\
&= \mathbb{E}_{\vec{x} \sim \{\pm 1\}^n} \left[ \frac{1 - f(\vec{x})f(\vec{x}^{\otimes i})}{2} \right] \\
&= \frac{1}{2} - \frac{1}{2} \mathbb{E}_{\vec{x}} \left( \sum_S \hat{f}(S) \chi_S(\vec{x}) \right) \left( \sum_T \hat{f}(T) \chi_T(\vec{x}^{\otimes i}) \right) \\
&= \frac{1}{2} - \frac{1}{2} \sum_{S,T} \hat{f}(S) \hat{f}(T) \mathbb{E}_{\vec{x}} \chi_S(\vec{x}) \chi_T(\vec{x}^{\otimes i}) \\
&= \frac{1}{2} - \frac{1}{2} \sum_{i \in S} \hat{f}(S) - \frac{1}{2} \sum_{i \notin S} \hat{f}(S)^2 \\
&= \sum_{i \in S} \hat{f}(S)^2
\end{aligned}$$

**Example 5.77.** Let  $f$  be the majority function over  $n = 2k + 1$  variables. Then

$$\text{Inf}_i(f) = \Pr_{\vec{x} \sim \{\pm 1\}^n} [\text{maj}(\vec{x}) \neq \text{maj}(\vec{x}^{\otimes i})] = \frac{\binom{2k}{k}}{2^{2k}} = \frac{(2k)!}{2^{2k}(k!)^2} \approx \frac{\sqrt{2\pi(2k)} \cdot \left(\frac{2k}{e}\right)^{2k}}{2^{2k} (2\pi k) \left(\frac{k}{e}\right)^{2k}} = \frac{\sqrt{4\pi k}}{2\pi k} = \Theta$$

**Theorem 5.78 (Majoiry is Stablest).** Let  $\rho \in (-1, 0)$ ,  $\varepsilon > 0$ . Then  $\exists \tau = \tau(\rho, \varepsilon) > 0$  such that  $\forall f : \{\pm 1\} \rightarrow \{\pm 1\}$ , if  $\text{Inf}_i(f) \leq \tau$ ,  $\forall i$ , then

$$\mathbb{S}_\rho(f) > 1 - \frac{2 \arccos \rho}{\pi} - \varepsilon$$

Therefore,

$$\Pr[f(\vec{x}) \neq f(\vec{x}\vec{\mu})] = \frac{1}{2} - \frac{1}{2} \mathbb{S}_\rho(f) < \frac{\arccos \rho}{\pi} + \frac{\varepsilon}{2}$$

□

**Theorem 5.79.** Let  $\rho \in (-1, 0)$ ,  $\varepsilon > 0$ . There exists  $\tau = \tau(\rho, \varepsilon) > 0$ ,  $C = C(\rho, \varepsilon) > 0$



such that  $f : \{\pm 1\}^n \rightarrow [-1, 1]$  with  $\max_i \{\text{Inf}_i^{\leq C} C(f)\} \leq \tau$ , we have

$$\mathbb{S}_\rho(f) \geq 1 - \frac{2 \arccos \rho}{\pi} - \varepsilon$$

**Definition 5.80.**  $\text{Inf}_i \leq C(F) = \sum_{S \ni i, |S| \leq C} \hat{f}(S)^2$

**Claim 5.9.1.**

$$\sum_{i \in [n]} \text{Inf}_i^{\leq C} f = \sum_{S, |S| \leq C} \hat{f}(S)^2 \cdot |S| \leq C \cdot \sum_S \hat{f}(S)^2 \leq C$$

Therefore,  $|\{i : \text{Inf}_i^{\leq C}(f) > \tau\}| < \frac{C}{\tau}$ .

i.e. If  $\Pr[\text{f pass dict test}] > \frac{\arccos \rho}{\pi} + \frac{\varepsilon}{2}$  then  $1 \leq |\{i : \text{Inf}_i^{\leq C}(f) > \tau\}| < \frac{C}{\tau}$ .

## 5.10 The Entire Proof

Unique Label Cover Game: Constraint graph  $G = (U, V, E)$  left-regular. Alphabet  $\Sigma = [K]$ . Constraints  $\forall e = (u, v) \in E, \pi_e : [K] \leftrightarrow [K]$ . Prover: provide  $\sigma : U \cup V \rightarrow [K]$ . Verifier: Sample  $(u, v) \sim E$ , test  $\pi_{uv}(\sigma(v)) = \sigma(u)$ .

Unique Games Conjecture states that  $\forall \delta > 0, \exists K = K(\delta)$  s.t.  
Gap-ULC( $K$ ) $_{1-\delta, \delta} \notin P$ .

Reduction to MaxCut:

# Index

$\alpha$ -approximation algorithm, [34](#)

3-CNF, [34](#)

arborescence, [13](#)

Bellman Equation, [15](#)

c vs. s Decision Problem, [35](#)

CNF, [34](#)

cut, [10](#)

cutset, [10](#)

fundamental cut, [10](#)

Fundamental cycle, [10](#)

gap ratio, [42](#)

hyperparameter, [16](#)

Integrality Gap (IG) instance, [42](#)

Laplacian, [76](#)

Markov Inequality, [40](#)

NP-complete, [47](#)

NP-hard, [47](#)

polynomial reduction, [47](#)

spanning tree, [10](#)

st-flow, [21](#)

Unique Label Cover Game, [85](#)

# Important Theorems

2.2	Theorem	7
2.6	Theorem (Invariant)	8
2.10	Theorem (Cayley Theorem)	10
2.11	Theorem	11
2.18	Theorem	14
4.2	Theorem	22
4.4	Theorem	24
4.11	Theorem (Hall’s Theorem)	27
4.14	Theorem (Menger’s Theorem)	28
4.25	Theorem	33
5.2	Theorem	35
5.11	Theorem (Markov Inequality)	40
5.12	Theorem	41
5.13	Theorem (Chernoff Bound)	44
5.14	Theorem (Cock-Levin)	47
5.15	Theorem (Max-Coverage)	48
5.16	Theorem (PCP theorem)	49
5.18	Theorem (PCP theorem)	49
5.21	Theorem (PCP theorem)	50
5.22	Proposition	51
5.23	Theorem	52
5.26	Theorem (not quite true)	53
5.27	Theorem	54
5.29	Theorem	54
5.35	Theorem	58

5.38 Theorem . . . . .	58
5.39 Theorem . . . . .	60
5.41 Theorem (Low-Diameter Decomposition, LDD) . . . . .	60
5.43 Theorem . . . . .	62
5.44 Theorem . . . . .	63
5.48 Theorem . . . . .	65
5.52 Proposition (Fourier expansion) . . . . .	67
5.53 Theorem (Parseval's) . . . . .	67
5.61 Theorem (Bourgain' 1985) . . . . .	73
5.62 Theorem (LLR 1995) . . . . .	73
5.63 Theorem . . . . .	74
5.64 Theorem (Low-Diameter Decomposition) . . . . .	74
5.66 Theorem . . . . .	79
5.74 Theorem (High-Dim Berry Esseen) . . . . .	87
5.78 Theorem (Majoiry is Stablest) . . . . .	88

## Important Examples

1.1 Example (Task Assignment) . . . . .	4
1.7 Example (Task machine) . . . . .	5
2.1 Example (Interval Scheduling) . . . . .	6
2.3 Example (Interval Partitioning) . . . . .	7
2.5 Example (Single-Source Shortest Path(SSSP)) . . . . .	8
2.8 Example (Minimum Spanning Tree (MST)) . . . . .	10
2.16 Example (Minimum Arborescence) . . . . .	13
3.1 Example (Weighted Interval Scheduling) . . . . .	15
3.2 Example (Least Square) . . . . .	16

3.3	Example (Segmented Least Square) . . . . .	16
3.4	Example (Knapsack Problem) . . . . .	16
3.6	Example (RNA Secondary Structure) . . . . .	19
3.7	Example . . . . .	19
3.8	Example (Matrix Multiplication) . . . . .	21
4.1	Example . . . . .	21
4.10	Example (Bipartite Matching) . . . . .	27
4.13	Example (Network Connectivity) . . . . .	28
4.15	Example . . . . .	28
4.17	Example (Survey Design) . . . . .	29
4.18	Example (Airline Scheduling) . . . . .	30
4.20	Example (Image Segmentation) . . . . .	30
4.22	Example (Project Selection) . . . . .	31
4.24	Example . . . . .	32
5.1	Example (3-CNF) . . . . .	34
5.5	Example (Set-Cover) . . . . .	36
5.10	Example (Weighted Min Set-cover) . . . . .	38
5.34	Example (Multicut) . . . . .	58
5.37	Example (Vertex Cover) . . . . .	58
5.42	Example (Max-cut) . . . . .	61
5.50	Example (Instance) . . . . .	65
5.54	Example (3-Coloring) . . . . .	69
5.55	Example (Min-3-Coloring) . . . . .	69
5.58	Example (Sparsest Cut) . . . . .	71
5.77	Example . . . . .	88